УДК 681.324

К.В. Дема, М.А. Волк, М.А. Филимончук

Харьковский национальный университет радиоэлектроники, Харьков

АНАЛИЗ АРХИТЕКТУР И ХАРАКТЕРИСТИК СОВРЕМЕННЫХ ПЛАНИРОВЩИКОВ ЗАДАНИЙ В GRID-СИСТЕМАХ

В работе рассматриваются принципы работы и функциональные требования к планировщикам заданий GRID-систем. Рассматривается классификация алгоритмов планирования. Проводится анализ архитектурных особенностей современных метапланировщиков заданий GRID-систем.

Ключевые слова: *GRID-система, планировщик заданий, алгоритм распределения.*

Введение

В современной информационной сфере все большее распространение получают GRID системы, которые оказались востребованными для решения как научных, так и коммерческих задач. Исследования и разработки в области объединения разнородных ресурсов и создания мощных вычислительных систем проводятся в рамках работы научных институтов и университетов мира, а также ведущими компаниями в сфере IT технологий, такими как IBM, Cluster Resources, Sun. Созданию, внедрению и использованию GRID систем сопутствует ряд открытых проблем, среди которых актуальной задачей является оптимальное планирование заявок и ресурсов с целью достижения максимально эффективной загрузки вычислительных комплексов. GRID системы – системы, позволяющие интегрировать гетерогенные вычислительные ресурсы, распределенные по всему миру, для решения сложных научных и инженерных задач, предъявляющих большие требования к производительности исполняющих узлов системы. Объединение разнородных ресурсов осуществляется за счет построения виртуальной организации GRID-систем – промежуточного уровня абстракции между пользователем и непосредственно исполнительными узлами системы. Общая концепция виртуальной организации GRID подробно раскрыта в статье Яна Фостера «Анатомия Грид» [1]. В своей работе рассматривается целостная открытая архитектура GRID систем, проблемы аутентификации, авторизации, обнаружения и упорядоченной организации ресурсов. Виртуальная организация GRID-систем представляет собой набор программных сервисов (Grid Middleware – GMW), реализующих функции авторизации, безопасности, управления ресурсами и потоками задач. Данный подход позволяет осуществить виртуализацию и масштабирование вычислительных ресурсов. Планировщики задач (также именуемые метапланировщиками или GRID брокерами) GRID-систем – неотъемлемые компоненты виртуальной организации GRID, основной задачей которых является планирование выполнения множества задач на распределенных ресурсах.

Выполнение заданий в GRID представлено на рис. 1 и состоит из следующих шагов. Пользователь отправляет задание планировщику в виде запроса, составленного с помощью специального языка описания (1). Планировщик использует сервисы GRID для обнаружения доступных ресурсов, получения информации о ресурсах и их оценки (2). На основании полученной информации, планировщик определяет, где будет выполнено задание. Выполнение задания управляется специализированными службами GRID (3).

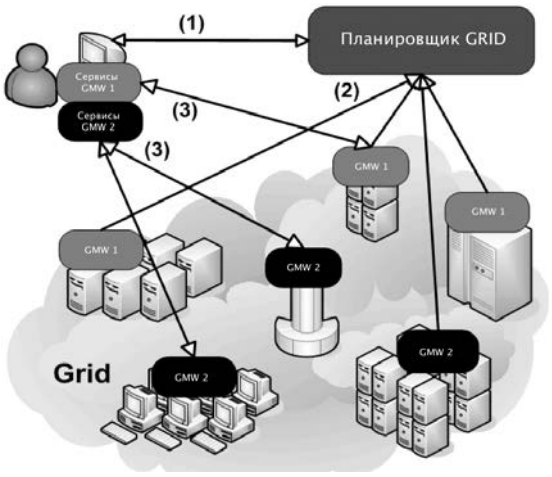

Рис. 1. Архитектура планирования и выполнения заданий GRID

GRID планирование значительно отличается от пакетного планирования задач в кластерных системах, так как в GRID системах информация о ресурсах быстро устаревает и ненадежна, что сильно усложняет реализацию задачи планирования заданий. В работе Е. Имамажича «Подход к планированию заданий в GRID системах на основе механизма сопоставления Condor-G» раскрываются основные требования к метапланировщикам GRID систем [2].

Целью данной статьи является рассмотрение ключевых особенностей построения планировщиков

GRID, анализ доступных и получивших широкое применение в современных реализациях GRID архитектур метапланировщиков, таких как MAUI Scheduler и PBS [3 – 7] с точки зрения классификации их функциональных и архитектурных решений.

Функциональные требования к планировщикам заданий GRID

Сложная комплексная архитектура построения GRID систем, цели и задачи работы GRID предъявляют ряд важных требований к планировщикам заданий[2]. Во-первых, планировщик заданий GRID должен поддерживать обработку различных типов поступающих заявок. Базовыми видами заданий являются последовательные (serial jobs) и параллельные (parallel jobs) задания. Последовательные задачи представляют собой приложения, требующие один процессор для выполнения, когда параллельные запрашивают множество процессоров для исполнения своего кода. Более сложными типами заданий являются массивы задач и потоки задач. Массив задач обычно представляет собой серию запусков одного и того же приложения с разными параметрами. Поток задач – набор разных приложений, работа которых зависит от состояния выполнения или конечного результата других приложений в этом наборе.

GRID планировщик должен поддерживать ряд функциональных механизмов, ставших стандартами в построении GRID систем [3]:

1. Контрольные точки – возможность сохранять текущее состояние и окружение задачи, для дальнейшего восстановления или перезапуска ее работы;
2. Остановка выполнения задачи для запуска другой с большим приоритетом;
3. Миграция заданий – возможность динамически переносить выполнение задачи с одного ресурса на другой;
4. Перепланировка заданий – необходима, когда выполнение текущего задания происходит с ошибками на специфичных ресурсах;
5. Устойчивость к ошибкам – возможность восстановления процесса выполнения от ошибок, возникающих на ресурсе или в приложении;
6. Предварительное резервирование – возможность заранее резервировать временной промежуток для выполнения определенных задач.

Планировщик должен иметь возможность назначать приоритеты заданиям и предоставлять такую функциональность пользователю для определения более важных задач среди набора заданий. Пользователь должен иметь возможность оценить приоритеты для доступных ресурсов и запросить специальные параметры, такие как программные библиотеки, совместимость оборудования. Планировщик так же должен предоставлять пользователю возможность применения собственных алгоритмов планирования и поддерживать реализации наиболее распространенных алгоритмов планирования. Учет местоположения массивов данных, необходимых задачам, и их передвижения по сети является важной функцией. Так, одним из алгоритмов размещения заданий может быть запуск их на ресурсах, близких к местам хранения данных, что позволит сократить задержки на передачу данных по сети.

Классификация алгоритмов планирования в GRID системах

Планирование заданий в GRID системах, как отмечает существует на двух уровнях: локальном и глобальном [4]. Локальное планирование осуществляется непосредственно в узлах GRID системы – в кластерах, сетях или локальных ЭВМ – и управляется системными администраторами или владельцами этих систем. В этом случае планировщик работает с четко заданным количеством ресурсов, обладает полной информацией об их состоянии и имеет возможность контролировать ресурсы. Задачи, которые решает такой сервис, сводятся к тому, как отдельная резидентная программа будет выполнена на отдельном процессоре. Метапланировщики GRID работают на глобальном уровне и используют информацию обо всех системах и их загруженности в целом, не имея при этом возможности контролировать процесс выполнения задания на конкретных удаленных ресурсах. На глобальном уровне могут работать как один, так и несколько планировщиков, что соответствует одной из схем планирования – централизованной или децентрализованной.

Централизованное планирование проще реализуется, но, из-за большой нагрузки, планировщик может столкнуться с проблемой нехватки производительности и стать «узким местом» системы. При децентрализованной схеме работы, планировщики могут работать с кооперацией или без кооперации. При кооперации, планировщики согласуют решения о запуске задания между собой, когда в альтернативном варианте каждый планировщик отвечает лишь за распределение своей части задач, но все планировщики при этом работают с целью эффективной загрузки системы. Глобальный уровень включает в себя две стратегии планирования: статическое и динамическое планирование. Статическое планирование предполагает, что на выбор ресурса для выполнения задания влияет информация о состоянии всех ресурсов до начала исполнения приложения. Динамическое планирование предполагает изменение расположения и состояния задания во время исполнения на основании новой информации о ресурсах, то есть «на лету». При принятии решения о распределении задач на ресурсы, могут применяться два способа оценки: оптимальный и субоптимальный. Оптимальный способ подразумевает, что известна информация обо всех задачах и ресурсах, а решение может быть принято на основе значения функции какого либо критерия оценки ресурса. Но в реальных GRID системах тяжело предсказать поведение исполняемых приложений и

параметров входного потока заявок, поэтому для более эффективного использования ресурсов часто используют субоптимальные алгоритмы. Субоптимальные алгоритмы подразделяются на приближенные и эвристические методы. Приближенные методы используют общие алгоритмы планирования, но не осуществляют поиск оптимального решения во всем пространстве возможных, а останавливаются при первом “хорошем” найденном решении. Эвристические алгоритмы позволяют сделать наиболее реальную оценку для решения задачи планирования, даже в тех ситуациях, когда не существует одного оптимального решения. Работа таких алгоритмов основываются на экспериментальных опытах в реальных системах или моделировании.

Обзор существующих реализаций планировщиков GRID

MAUI Scheduler

Планировщик MAUI является одним из самых популярных и эффективных метапланировщиков GRID и используется во многих реализациях GRID систем, таких как IBM Tivoli[5] и Moab Workload Manager [6]. Ключевой особенностью данного планировщика является система резервации времени для заданий.[7]. Система сортирует очередь заданий по их приоритетам, отправляет на выполнение первую задачу с наибольшим приоритетом, резервируя время для следующей задачи в очереди. После того, как выстроен план запуска задач с большими приоритетами, с помощью механизма Backfill, MAUI пытается найти в оставшихся временных промежутках места для задач с меньшими приоритетами. Такой алгоритм позволяет гарантированно обслужить задачи с высоким приоритетом и обеспечить приемлемый оборот в системе мелких заданий. Рассмотрим более подробно шаги работы планировщика. Обработка входящих запросов проходит итеративно. Каждая итерация состоит из двух периодов – периода планирования и таймаута.

Период планирования подразделяется на два этапа: этап планирования очереди приоритетов и этап заполнения временных промежутков (Backfill scheduling). Во время фазы планирования очереди приоритетов MAUI определяет, какие задания могут быть выполнены на ресурсах. В этом процессе учитываются политики легитимности (fairness policies), состояние задания, ограничения, накладываемые системой GRID, и другие факторы. Полученному списку заданий, удовлетворяющих условиям запуска на выполнение, назначаются приоритеты, список сортируется. Планировщик обходит отсортированную очередь и пытается распределить задания в соответствии с приоритетами. Когда планировщик обнаруживает первое задание, которое не может быть запущенно немедленно, он резервирует самый ближайший возможный отрезок времени для запуска этого задания, который определяется в рамках требований задания по времени исполнения. Также резервируется набор ресурсов для выполнения этого задания. Резервирование состоит из списка ресурсов, владельца и даты начала и окончания резервирования. Таким образом, ресурсы могут быть использованы для других целей в остальное время. После распределения заданий, которые могут быть запущены, и резервации времени для ожидающих заданий с большими значениями приоритета, планировщик переходит ко второй фазе – фазе заполнения неиспользованных временных промежутков (Backfill phase). Концепция второй фазы планирования состоит в том, чтобы распределить те задания, которые не повлияют на время выполнения задач, находящихся впереди в очереди приоритетов. Существует несколько вариантов реализации данного алгоритма. По умолчанию планировщик использует «свободную» политику, руководствуясь правилом: «распределить те задания, которые не задержат выполнение задания с наибольшим приоритетом, ожидающего в очереди». Однако можно настроить более жесткие условия распределения заданий механизмом backfill. Фаза заполнения промежутков состоит из двух фаз, основанных на применении политик легитимности. Все политики легитимности имеют «жесткое» и «мягкое» значение. Например, политики, контролирующие максимальное возможное число запущенных заданий для пользователя, могут иметь «жесткое» значение, равное 4, и «мягкое», равное 2. Администратор в таком случае устанавливает ограничение: «Пользователю разрешено запустить 2 задания одновременно. Если есть простаивающие ресурсы, которые не могут быть использованы другими заданиями, то возможен запуск 4 заданий»[8]. Во время первой фазы применяются более жесткие ограничивающие значения политик, а во время второй фазы происходит поиск возможных решений для оставшихся ресурсов на основе менее строгих параметров. Фаза заполнения временных промежутков организует процесс поиска заданий для распределения с помощью окон (Backfill Window). На момент завершения фазы назначения приоритетов очереди, система будет содержать набор простаивающих ресурсов в следующем виде (табл. 1):

Таблица 1

Представление свободных ресурсов в планировщике заданий MAUI

Имя узла	Время простоя
Node A	1000 seconds
Node B	300 seconds
Node C	300 seconds
Node D	500 seconds
Node E	200 seconds
Node F	NO LIMIT
Node G	NO LIMIT
Node H	300 seconds

Механизм backfill на основе этой информации организует все ресурсы в специальные окна в виде списка:

[1] 8 Узлов доступно на 200 seconds
[2] 7 Узлов доступно на 300 seconds
[3] 4 Узлов доступно на 500 seconds
[4] 3 Узлов доступно на 1000 seconds
[5] 2 Узлов доступно без ограничения.

По умолчанию планировщик MAUI сначала попытается заполнить первое окно, то есть распределить все задания, которые требуют 8 или менее ресурсов и время выполнения 200 или менее секунд. Если остаются свободные ресурсы после заполнения первого окна, планировщик ищет задания, совпадающие по требованиям с параметрами второго окна. После каждого распределения информация о параметрах ресурсов в «окне» обновляется. То есть, если в первом окне были распределены задания на время выполнения 200 секунд, доступное время других ресурсов во втором и других «окнах» изменится. Фактически, второе окно не будет создано, пока не заполнится первое. Планировщик использует простой алгоритм первого совпадения (first fit) для поиска заданий для окна. Однако доступны и другие алгоритмы, такие как алгоритм наилучшего соответствия (best fit), «жадный» алгоритм и другие. Выбрать алгоритм можно в настройках планировщика. Доступна конфигурация политики составления окна, используя подход поиска ресурса с наибольшим временем простоя (Longest-Time-First) вместо поиска наибольшего количества ресурсов с приблизительно одинаковым временем простоя (Most-Nodes-First).

Condor-G

Condor [9] – специализированная система управления заданиями и ресурсами, которая содержит в себе такие механизмы, как управление и мониторинг заданий и ресурсов, планирование и приоритизация заданий. Т. Таненбаум в работе «Condor и GRID» предоставляет детализированное описание механизмов планировщика и его интеграцию в GRID системы разных типов. Общая схема работы проста: пользователь отправляет задание, Condor планирует его выполнение, распределяет и дает возможность управлять процессом исполнения задания. Однако Condor реализует ряд уникальных механизмов, которые позволяют системе успешно функционировать в таких областях как высокопроизводительные длительные вычисления и вычисления с максимальным использованием свободного ресурсного времени. Condor содержит набор функциональных возможностей:

1. Механизм объявлений классификации требований. Гибкий механизм для сопоставления запросов в виде заданий и доступных ресурсов, применения политик и алгоритмов.

2. Контрольные точки и миграция, – позволяют сохранять состояния выполнения заданий и перезапускать их на других ресурсах в зависимости от текущей ситуации в системе.

3. Удаленные системные вызовы. Система перенаправляет вызовы ввода/вывода на машину, с которой было отправлено задание, что позволяет не заботится о переносе данных.

Condor может не только работать с кластерами, но и с десктопными машинами, используя их процессорное время, например, когда клавиатура и мышь не активны. Condor-G представляет собой синтез систем Condor и Globus. Condor отвечает за добавление, планирование, управление заданиями, восстановление ошибок, а Globus реализует остальные сервисы GRID middleware компоненты. При планировке заданий используется специальный сервер Matchmaker, который сопоставляет свободные ресурсы и поступившие заявки на обработку. Общая схема работы представлена на рис. 2.

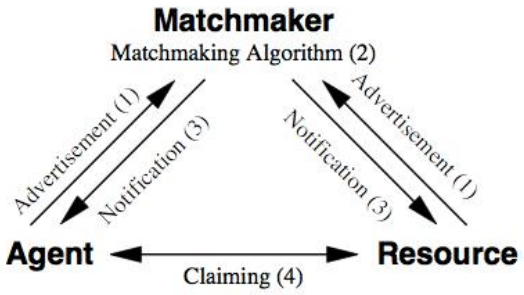

Рис. 2. Схема работы планировщика Condor-G

Ресурсы и клиенты предоставляют информацию о себе(1), Matchmaker принимает решение о сопоставлении здания и ресурса, отсылает уведомления агенту и ресурсу (3), агент и ресурс взаимодействуют друг с другом для запуска задания.

Процедура нахождения соответствия (matchmaking) требует четырех шагов. На первом шаге, агент (пользователь) и ресурсы предоставляют информацию (classified advertisement – CA) о требованиях и характеристиках своих систем и заданий планировщику. На втором шаге, matchmaker формирует пары значений агент-ресурс, удовлетворяющие предъявленным CA. На третьем шаге matchmaker уведомляет пару агент-ресурс об установленном соответствии. После выполнения перечисленных действий планировщик не несет ответственности за действия агента и ресурса. На четвертом шаге агент и ресурс устанавливают соединение и начинают передачу данных.

CA – набор уникально поименованных выражений, представленных в виде структуры данных. Каждое выражение называется атрибутом, каждый атрибут имеет имя и значение. Значение может быть целым числом, строкой или числом с плавающей точкой и т.д. Выражение содержит логические и арифметические операторы. Пример описания приведен на рис. 3, 4.

```
Job ClassAd
[
MyType = "Job"
TargetType = "Machine"
Requirements =
((other.Arch=="INTEL" &&
other.OpSys=="LINUX")
&& other.Disk > my.DiskUsage)
Rank = (Memory * 10000) + KFlops
Cmd = "/home/tannenba/bin/sim-exe"
Department = "CompSci"
Owner = "tannenba"
DiskUsage = 6000
]
```

Рис. 3. Пример описания требований агента

```
Machine ClassAd
[
MyType = "Machine"
TargetType = "Job"
Machine = "nostos.cs.wisc.edu"
Requirements =
(LoadAvg <= 0.300000) &&
(KeyboardIdle > (15 * 60))
Rank = other.Department==self.Department
Arch = "INTEL"
OpSys = "LINUX"
Disk = 3076076
]
```

Рис. 4. Пример описания характеристик ресурса

Condor придает особое значение атрибутам Requirements (требования) и Rank (ранг). Требования характеризуют ограничения, накладываемые агентом на ресурс, а ранг – приоритет удовлетворения тех или иных требований. Данные два атрибута являются необходимыми для нахождения соответствия между агентом и ресурсом. В примере, изображенном на рис. 3, задание требует Intel Linux систему для запуска со свободной памятью в 6 мегабайт. Среди всех ресурсов, удовлетворяющих перечисленным требованиям, агент отдает приоритет ресурсам с наибольшим количеством свободной памяти и хорошей производительностью вычислений чисел с плавающей точкой. Описание ресурса показывает, что клавиатура простаивает в течение 15 минут и ЭВМ запустит задание на выполнение, только если его общая загрузка процессора небольшая. Значение ранга говорит о том, что для ЭВМ задания, отправленные из собственного отделения, приоритетнее других. Condor-G может осуществлять планирование «вокруг» других планировщиков. Удаленные ресурсы управляются своими локальными планировщиками, поэтому, отправляя задание в очередь ресурса, Condor-G не может контролировать планирование задания внутри этого ресурса. Но если удаленный планировщик предоставит свое расписание в виде CA, то Condor-G может принимать более успешные решения по распределению заданий. Существует другой способ планирования – планирование «внутри». Суть заключается в том, что Condor-G предлагает список ресурсов агенту. При получении подтверждения, агент получает в свое владение эти ресурсы и осуществляет планирование в их пределах самостоятельно. Такой подход часто используется при распределении параллельных задач.

Condor реализует систему контрольных точек, во время формирования которых происходит сохранение состояния задания, после чего задание может быть снято с ресурса и перераспределено на других машинах. Для сохранения состояния используется отдельный сервер. Контрольные точки бывают двух видов и также подлежат планированию, как и задания. Первый вид – приоритетные контрольные точки. Планировщик составляет расписание для каждого приложения и резервирует пропускную способность сети под передачу данных приложения. Когда приходит назначенное время, планировщик дает команду приложению начать передачу сохраненных данных. Второй вид – периодические контрольные точки. Планировщик инициирует периодическое сохранение, когда загрузка сети незначительна. При этом данные сериализуются и передача выполняется быстрее. Периодическое сохранение не может быть выполнено, если в это же время запланирован запуск приоритетной контрольной точки.

PBS (Portable Batch System)

PBS – система управления и распределения заданий, в составе которой существует собственный планировщик [10]. Система использует разнообразные настраиваемые политики управления заданиями для распределения задач. Планировщик позволяет создавать учетные записи пользователей, группировать ресурсы, задавать ограничения для поступающих заданий, осуществлять выгрузку и перераспределение задач, назначать приоритеты, применять политику «справедливого» разделения ресурсов. Переодически происходит сбор сведений о ресурсах, фиксируется информация о количестве процессоров, свободной памяти, архитектуре операционной системы. Эти данные необходимы для сопоставления с требованиями поступающих заявок. PBS использует несколько очередей, соответствующих набору приоритетов, и сортирует их по убыванию приоритетов. Задания в очереди со специальным значением приоритета preemt_queue_prio и выше инициирует выгрузку выполняемых заданий из очередей с меньшими приоритетами. В самой же очереди задания сортируются по времени поступления в очередь. Задания имеют предел ожидания, равный по умолчанию 24 часа. Чем дольше ждет задание в очереди, тем больше становится его приоритет выполнения.

PBS позволяет объединять ресурсы (vnodes) в специальные наборы виртуальных узлов (placement sets) по каким либо признакам на усмотрение адми-

нистратора. Такие наборы становятся объектами планирования и применения политик распределения заданий для балансирования нагрузки. С помощью таких наборов PBS реализует механизм предварительной резервации времени для входных задач. В настройках можно привязать какой-либо набор ресурсов к очереди, тогда задания из этой очереди будут распределены в пространстве своих виртуальных узлов. Такой механизм позволяет гибко описывать сложные вычислительные структуры и эффективно настраивать PBS. Планировщик опционально может осуществлять балансирование загрузки ресурса. Задавая числовой предел максимальной загрузки, планировщик не будет пытаться распределить задания, суммарная загрузка которых будет выше предела.

Система PBS позволяет осуществлять совместную работу нескольких PBS комплексов. Каждый такой комплекс содержит свой планировщик и набор очередей. При такой кооперации может осуществляться "peer scheduling" – планирование между комплексами PBS. Тогда задачи из очередей одних вычислительных структур могут выполняться в других вычислительных системах при условии, что они будут запущенны немедленно. PBS позволяет гибко настраивать связи между очередями, определять пути пересылки заданий в их пределах.

Выводы

Следует заметить, что ни одна реализация планировщиков GRID не соответствует всем предъявляемым требованиям. Наиболее эффективного способа планирования на данный момент не существует. Современные планировщики GRID либо предоставляют набор политик и простых алгоритмов, конфигурирование которых позволяет добиться наилучших результатов опытным путем, либо приводят задания и ресурсы к общему описанию, что упрощает сопоставление требований, однако усложняет общую настройку GRID системы. Приведенные в статье характеристики планировщиков фактически стали стандартом при создании эффективных моделей, однако не гарантируют максимальную эффективность. Большинство метапланировщиков имеют возможность подключать пользовательские алгоритмы для решения спектра узких задач и используют набор простых алгоритмов по умолчанию, которые на практике показывают лучшие результаты по сравнению с использованием усложненных алгоритмов [8].

Список литературы

1. Ian Foster, Carl Kesselman, Steven Tuecke "THE ANATOMY OF THE GRID" // International Journal of High Performance Computing Applications, Volume 15 Issue 3, August 2001. – P. 25-37.

2. Emir Imamagic, Branimir Radic, Dobrisa Dobrenic "An Approach to Grid Scheduling by Using Condor-G Matchmaking Mechanism" // Journal of Computing and Information Technology, Volume 14, Number 1, 2006. – P. 38-45.

3. Globus Toolkit Documentation. [Электронный ресурс]. – Режим доступа: http://www.globus.org/toolkit/

4. Dr. Inderveer Chana "Grid Scheduling Algorithm based on Dynamic Time Quantum"

5. Tivoli Workload Scheduler Documentation. [Электронны ресурс]. – Режима доступа: http://www-01.ibm.com/software/tivoli/

6. Moab Workload Manager Documentation. [Электронный ресурс]. – Режим доступа: http://www. adaptive-computing.com/

7. Brett Bode, David M. Halstead "The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters" // ALS'00 Proceedings of the 4th annual Linux Showcase & Conference – Volume 4.

8. Информационный портал ANU Supercomputer Facility. [Электронный ресурс]. – Режим доступа: http://anusf.anu.edu.au/~dbs900/PBS/Maui/scheduling.html.

9. Douglas Thain, Todd Tannenbaum, and Miron Livny "Condor and the Grid" in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, Grid Computing: Making The Global Infrastructure a Reality, John Wiley, 2003.

10. Информационный портал PBS Works [Электронный ресурс]. – Режим доступа: http://www.pbsworks. com.

Поступила в редколлегию 2.03.2011

Рецензент: д-р техн. наук, проф. С.Г. Удовенко, Харьковский национальный университет радиоэлектроники, Харьков.

АНАЛІЗ АРХІТЕКТУР ТА ХАРАКТЕРИСТИК СУЧАСНИХ ПЛАНУВАЛЬНИКІВ ЗАВДАНЬ У GRID-СИСТЕМАХ

К.В. Дема, М.О. Волк, М.А. Филимончук

Розглядаються принципи роботи та функціональні вимоги до планувальників завдань GRID-систем. Розглядається класифікація алгоритмів планування. Проводиться аналіз архітектурних особливостей сучасних метапланувальників завдань GRID-систем.

Ключові слова: *GRID-система, планувальник завдань, алгоритм розподілу.*

ANALYSIS OF MODERN GRID SCHEDULING ARCHITECTURES AND CHARACTERISTICS

K.V. Dema, M.A. Volk, M.A. Filimonchuk

Is considered principles of working and functional requirements for GRID schedulers. Is considered a classification of scheduling algorithms. Analysis of architecture features of modern GRID schedulers is conducted.

Keywords: *GRID-system, planner of tasks, distributing algorithm.*