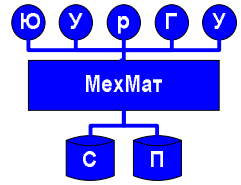


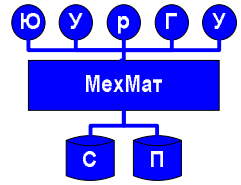
Грид технологии

Лекция 8
Планировщики и брокеры ресурсов. Система Condor

Радченко Глеб Игоревич, каф. СП, ЮУрГУ

http://dom.susu.ru/grid.htm

Содержание

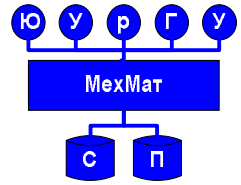


1. Планировщики и брокеры ресурсов
2. Система Condor
3. Архитектура Condor. Роли и процессы

1

Планировщики заданий

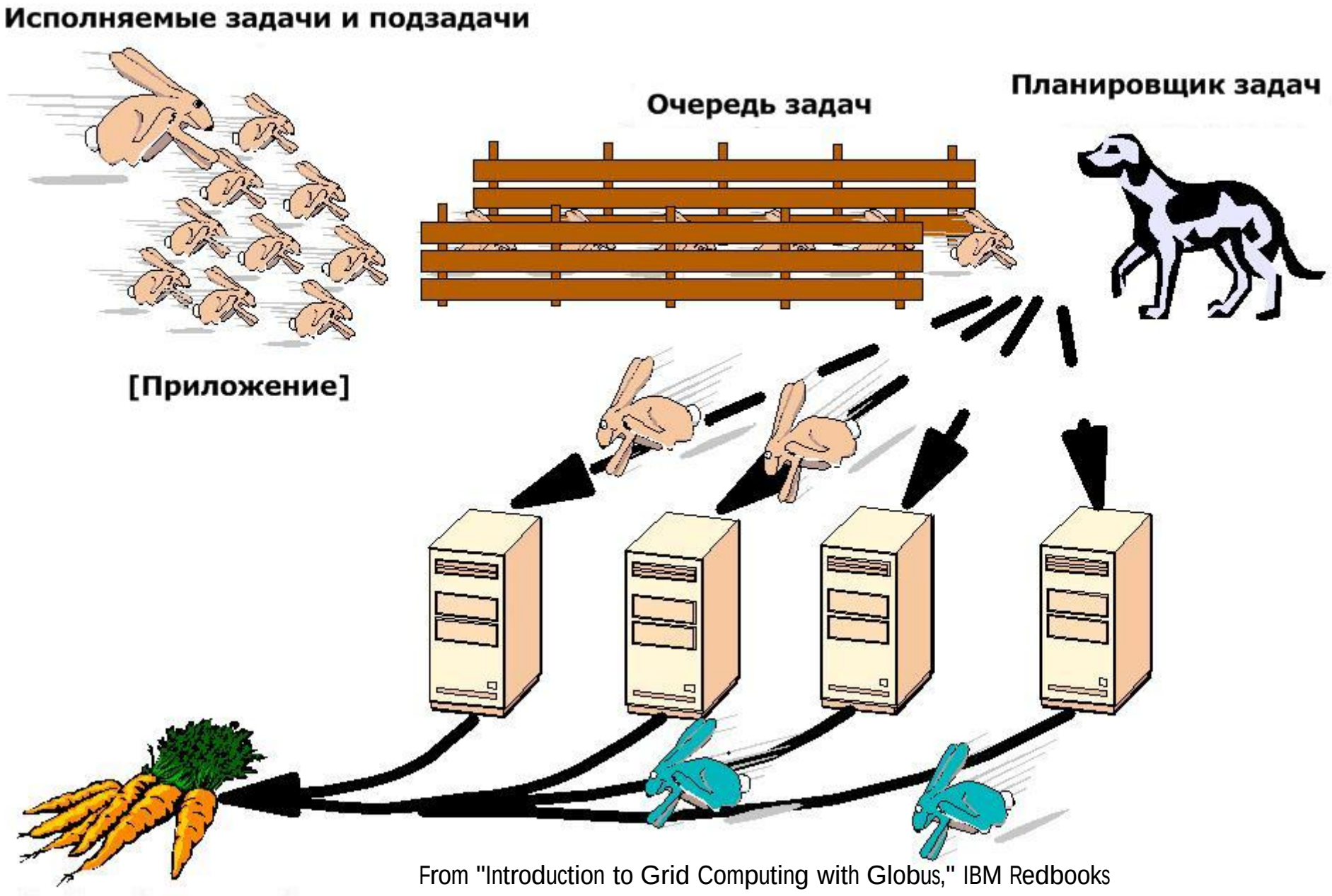
Планировщик заданий



- Менеджер задач передает принятые задачи *планировщику или брокеру задач.*

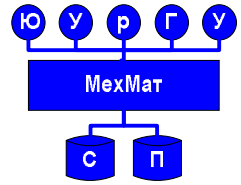
- Планировщик задач отправляет задачи на ресурсы, которые соответствуют необходимым требованиям и обеспечивают оптимальное время исполнения задачи.

Планировщик



From "Introduction to Grid Computing with Globus," IBM Redbooks

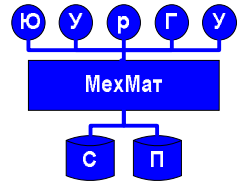
Исполнение задач GT4



Globus поддерживает следующие режимы постановки и исполнения задач:

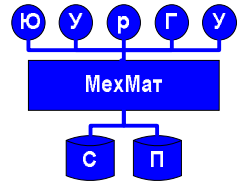
- Интерактивный (Interactive)
- Интерактивно-потоковый (Interactive-streaming)
- Пакетный (Batch)

Планировщик GT4 “Fork”



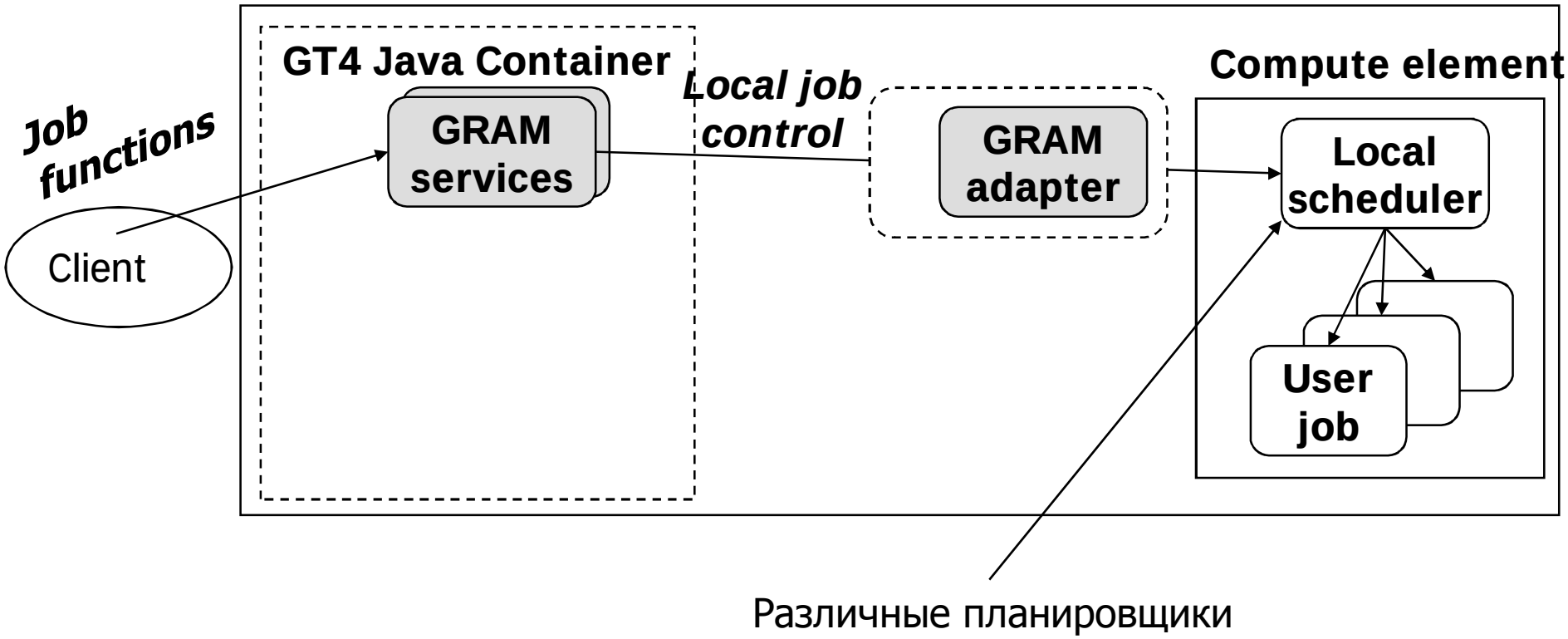
- При поступлении задачи пытается исполнить ее немедленно.
- Обеспечивает запуск и управление исполнением задачи на локальном узле, если задача не требует какого-либо специального ПО или параметров окружения.

Пакетное планирование



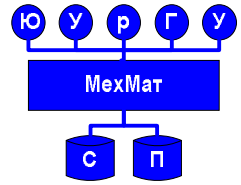
- Пакетный запуск: на узел производится постановка пакета задач, после чего планировщик самостоятельно занимается поиском и выделением ресурсов для каждой из задач. Пользователь в этом процессе не играет никакой роли. Как только решены все задания из пакета, планировщик завершает свою работу.

Взаимодействие между GRAM GT4 и локальным планировщиком



I Foster

Адаптеры планировщиков в GT 4

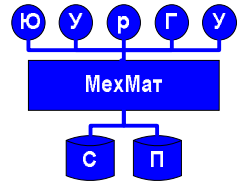


- PBS (Portable Batch System)
- Condor
- LSF (Load Sharing Facility)

Отдельно существует адаптер для:

- SGE (Sun Grid Engine)

Запуск заданий со внешними планировщиками

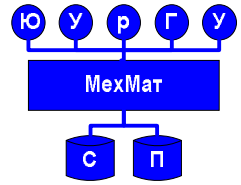


Примеры

globusrun-ws -Ft Condor on coit-grid02-4

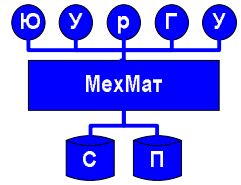
globusrun-ws -Ft SGE coit-grid01 & toralds.cis.uncw.edu

Задачи планировщика



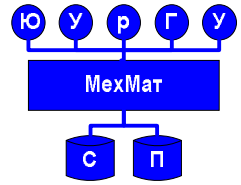
- Распределение задач
  - На основе загрузки и других характеристик машин, таких как свободное пространство, характеристики сети и т.п.
- Перераспределение данных для решения
  - Обеспечение репликации и передачи данных
  - Проверка правильности передачи данных

Задачи планировщика



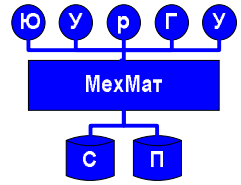
- Поддержка работы
  - Проверка ошибок и обеспечение контрольных точек
  - Мониторинг задач и процесса решения
  - Обеспечение качества (QoS: Quality of service)
- Безопасность
  - Аутентификация и авторизация удаленного пользователя для постановки задачи
- Устойчивость к ошибкам

Политики планирования



- Очередь (FIFO)
- Предпочтение определенному типу задач
- Предпочтение кратчайшей задаче
- Предпочтение наименьшему (наибольшему) объему памяти
- Равномерное распределение или предпочтение определенным пользователям
- Динамические политики
  - В зависимости от времени суток и нагрузки
  - Пользовательские политики

Предварительное Резервирование

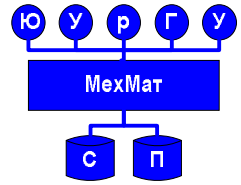


- Запрос определенных ресурсов или действий на будущее

- “Договор между службами, по которому определенные условия должны быть обеспечены к определенному моменту времени в будущем”

From: “The Grid 2, Blueprint for a New Computing Infrastructure,” I. Foster and C. Kesselman editors, Morgan Kaufmann, 2004.

Брокер ресурсов



- “Планировщик, оптимизирующий производительность отдельного ресурса.

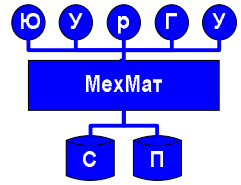
- Производительность может определяться различными критериями, например честность (обеспечение всех запросов на ресурсы) или утилизация (максимизация занятости ресурсов).”

From: “The Grid 2, Blueprint for a New Computing Infrastructure,” I. Foster and C. Kesselman editors, Morgan Kaufmann, 2004.

2

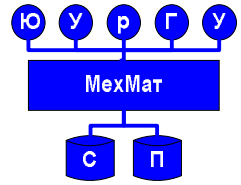
Система Condor

Condor



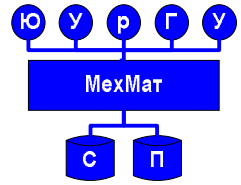
- Разработка ведется в рамках Condor Research Project в университете Wisconsin-Madison (США) с 1988 года
- Программное обеспечение и документация доступны бесплатно на Web-сайте проекта:
  - http://www.cs.wisc.edu/condor/
- Основная концепция – использование ресурсов простаивающих рабочих станций.

Condor



- Преобразует набор распределенных рабочих станций и кластеров в единую распределенную высокопроизводительную вычислительную среду.

Задача

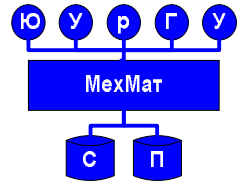


Вычислительный эксперимент: найти значения функции F(x,y,z) для

- 20 значений x;
- 10 значений y;
- 3 значения z.

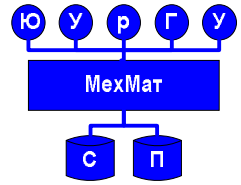
Итого 20 x 10 x 3 = 600 комбинаций.

Параметры задачи



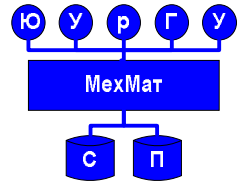
- На типовой рабочей станции вычисление F занимает 6 часов. Таким образом, на проведение всего эксперимента необходимо 600 x 6 = 3600 часов.
- Для вычисления F необходим средний объем оперативной памяти (256 Мб).
- F генерирует средний объем данных ввода/вывода:
  - (x,y,z) – порядка 5 Мб;
  - F(x,y,z) – порядка 50 Мб.

Подход Condor



- Эффективное использование доступных ресурсов, в первую очередь – простаивающих рабочих станций
- Ресурсы рабочих станций используются в среднем на 5-20%
- Объединение вычислительных мощностей простаивающих машин в *виртуальный кластер* (Condor pool)
  - делает использование существующих ресурсов более эффективными
  - расширяет число ресурсов доступных пользователю
  - позволяет управлять вычислительной средой с распределенным владением в отсутствие централизованного администрирования

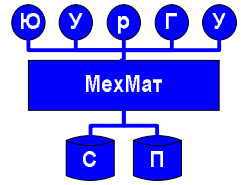
Система пакетной обработки (batch processing system)



- Прием заданий от пользователей
- Управление очередью заданий
- Распределение заданий между доступными ресурсам и планирование их выполнения в соответствии с заданной политикой
- Мониторинг выполнения задания
- Уведомление пользователя о результатах

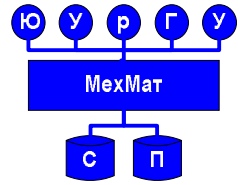
- Традиционные СПО - поддержка кластеров из *выделенных* машин
- Condor
  - Работает поверх сети из обычных настольных ПК и рабочих станций, на одной машине или выделенном кластере
  - Не требует централизованного администрирования
  - Работает в глобальных сетях
  - Автоматически определяет факт простоя машины и отправляет на нее задание
  - Автоматически осуществляет миграцию заданий

Задание в Condor



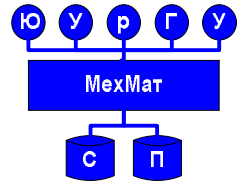
- Запуск заданного исполняемого кода (программы) на заданных входных данных
- В описании задания (Submit Description File) указываются
  - путь к программе
  - путь к входным данным
  - путь к месту, где следует разместить результаты вычислений (выходные данные)

Пример описания задания



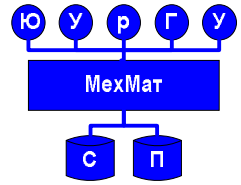

```
# Example condor_submit input file
# (Lines beginning with # are comments)
# NOTE: the words on the left side are not
#       case sensitive, but filenames are!
Universe   = vanilla
Executable = /home/frieda/condor/my_job.condor
Log        = my_job.log
Input      = my_job.stdin
Output     = my_job.stdout
Error      = my_job.stderr
Arguments  = -arg1 -arg2
InitialDir = /home/frieda/condor/run_1
Requirements = Memory >= 256 && Disk > 10000
Queue
```

Condor



- Включает в себя:
  - Поисковик ресурсов
  - Менеджер очереди задач
  - Планировщик
  - Механизм контрольных точек
  - Миграция задач

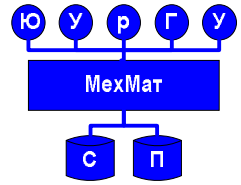
Condor



Рассчитан на то, чтобы решить задачу даже если:

- Отключается отдельный выч. узел
- Заканчивается дисковое пространство
- Не установлено требуемое программное обеспечение
- Узлы необходимы для других задач
- Узлы управляются другими системами
- Узлы распределены географически

Задача пользователя решена

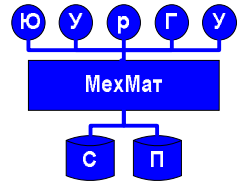


- С использованием 20 простаивающих по ночам машин компьютерного зала, пользователь может решить свою задачу за 3600 / 20 = 180 часов

3

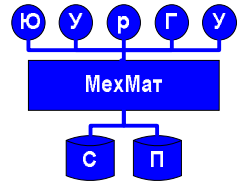
Архитектура Condor. Роли и процессы

Роли



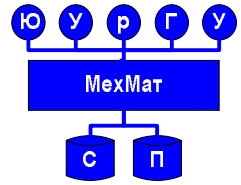
- Узлы в Condor могут исполнять 4 роли:
  - Центральный управляющий узел (Central Manager)
  - Узел постановки задач (Submit host)
  - Исполняющий узел (Execution host)
  - Сервер контрольных точек (Checkpoint server)

Центральный управляющий узел



- Брокер ресурсов пула.
- Следит за доступностью узлов, запущенных заданиях, распределяет задачи и т.д.
- В пуле только 1 центральный узел.

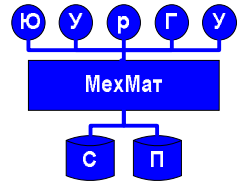
Узел постановки задач



- Узел, который обеспечивает постановку задач пулу.

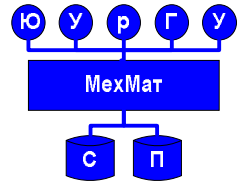
- В пуле должен быть как минимум один узел постановки задач (обычно более одного).

Исполняющий узел



- Узел на котором происходит решение поставленных задач.
- В пуле должен быть как минимум один исполняющий узел (обычно больше).

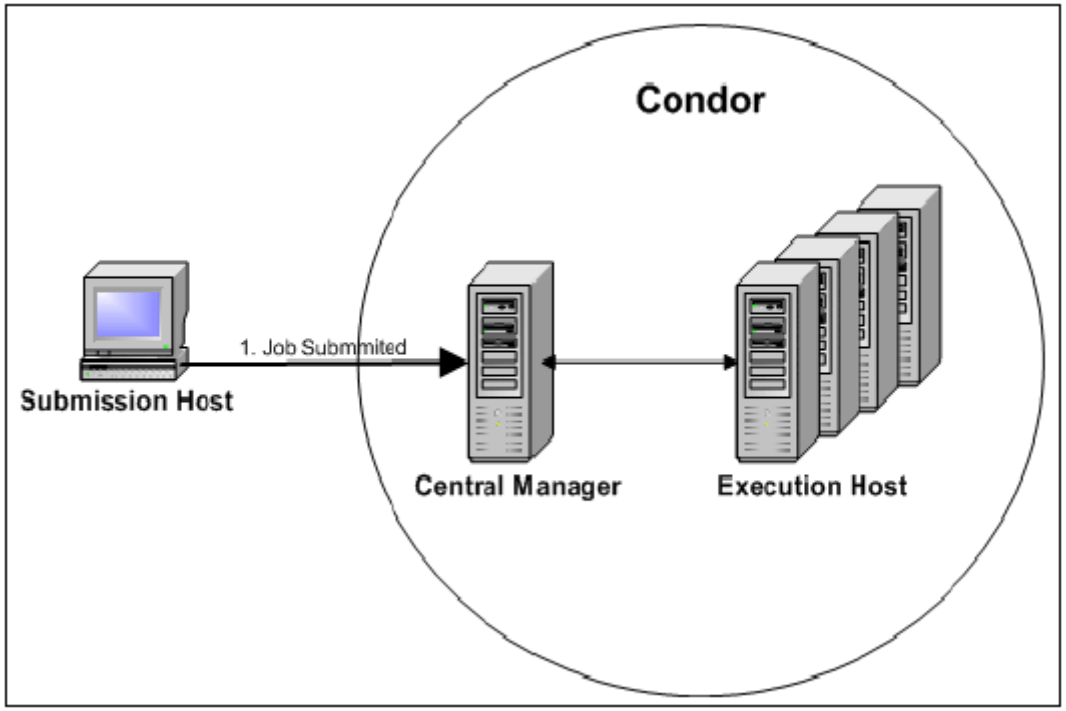
Сервер контрольных точек

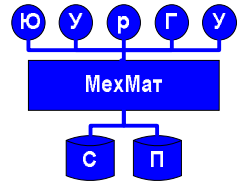


- Сервер, хранящий всю информацию о контрольных точках, генерируемых определенной задачей.
- В пуле может быть только один сервер контрольных точек.
- Можно отказаться от использования сервера контрольных точек.

Архитектура Condor

- Центральный менеджер (central manager)
- Выполняющие узлы (execution hosts)
- Запускающие узлы (submission hosts)
- + Сервер контрольных точек (checkpoint server)

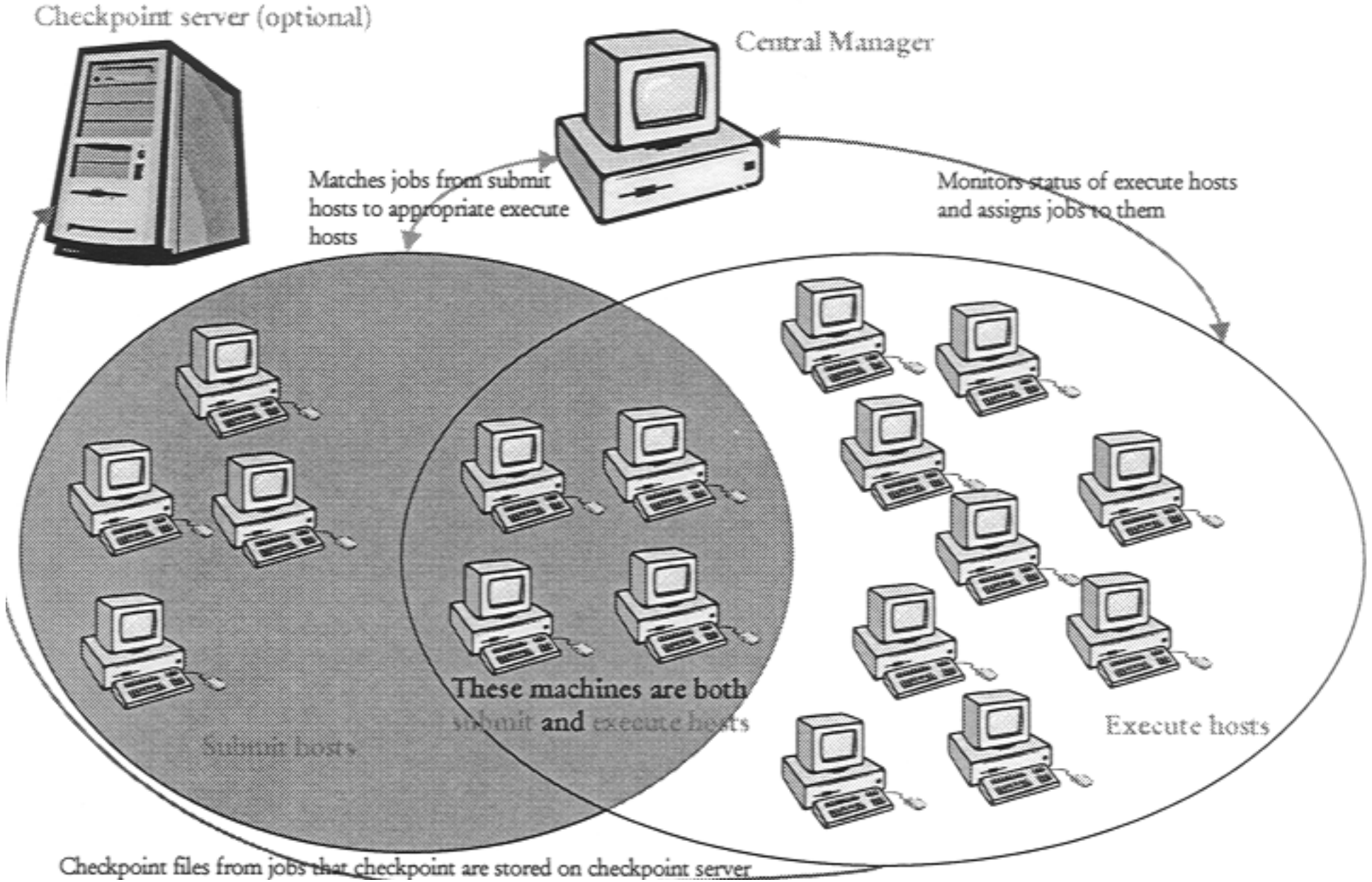




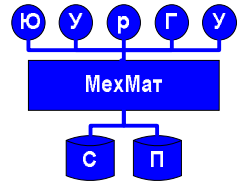
Пример конфигурации

- Центральный узел.
- Некоторые узлы – только для постановки задач.
- Некоторые узлы – только для исполнения задач.
- Некоторые узлы совмещают роли постановщика и исполнителя.

A typical Condor Pool

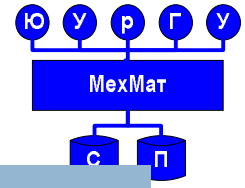


Типы заданий (universes)



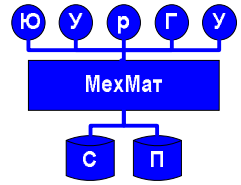
- Запуск обычных программ
  - Vanilla Universe (без перекомпиляции, C/C++)
  - Standard Universe (компоновка с библиотеками Condor)
    - Поддержка *удаленных системных вызовов*
    - Поддержка *контрольных точек*
- Java Universe (запуск программ на Java)
- Запуск параллельных программ
  - MPI Universe
  - PVM Universe
- Scheduler Universe
- Grid

Пример описания машины



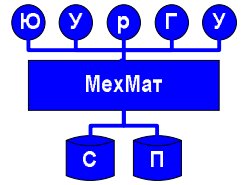

```
[
        Type = "Machine";
        Activity = "Idle";
        KeybrdIdle = '00:23:12'; // h:m:s
        Disk = 323.4m; // mbytes
        Memory = 64m; // mbytes
        State = "Unclaimed";
        LoadAvg = 0.042969;
        Mips = 104;
        Arch = "INTEL";
        OpSys = "SOLARIS251";
        KFlops = 21893;
        Name = "foo.cs.wisc.edu";
        ResearchGp = { "raman", "miron", "solomon" };
        Friends = { "calvin", "hobbes" };
        Untrusted = { "rival", "riffraff" };
        Rank = member(other.Owner, ResearchGp) ? 10 :
                  member(other.Owner, Friends) ? 1 : 0;
        Constraint = !member(other.Owner, Untrusted) && Rank>=10 ? true :
                  Rank>0 ? LoadAvg < 0.3 && KeybrdIdle>'00:15' :
                  DayTime()<'8:00' || DayTime()>'18:00'
]
```

Запуск пакета заданий



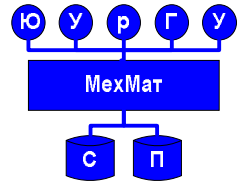

```
% condor_submit my_job.submit-file
Submitting job(s).
2 job(s) submitted to cluster 2.
% condor_q
-- Submitter: perdita.cs.wisc.edu : <128.105.165.34:1027> :
 ID      OWNER           SUBMITTED     RUN_TIME ST PRI SIZE CMD
  1.0    frieda          4/15 06:52  0+00:02:11 R  0   0.0  my_job
  2.0    frieda          4/15 06:56  0+00:00:00 I  0   0.0  my_job
  2.1    frieda          4/15 06:56  0+00:00:00 I  0   0.0  my_job
3 jobs; 2 idle, 1 running, 0 held
%
```

Пакет из 600 заданий



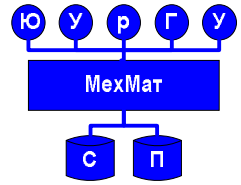

```
# Example condor_submit input file that defines
# a cluster of 600 jobs with different directories
Universe   = vanilla
Executable = my_job
Log        = my_job.log
Arguments  = -arg1 –arg2
Input      = my_job.stdin
Output     = my_job.stdout
Error      = my_job.stderr
InitialDir = run_$(Process)    ·run_0 … run_599
Queue 600                      ·Becomes job 3.0 … 3.599
```

Механизм ClassAd



- Используется для сопоставления спроса на ресурсы (заданий) и предложений ресурсов (машин)
- Задания могут включать требования и предпочтения к ресурсам
- Машины могут указывать требования и предпочтения к заданиям, которые они готовы выполнять
- Требования и предпочтения описываются при помощи достаточно гибких выражений, что позволяет системе адаптироваться к практически любой политике использования

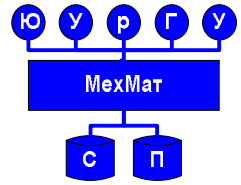
Требования задачи



Задачи могут определять требования к предоставляемому оборудованию:

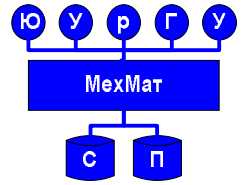
- Мне необходима платформа Linux/x86
- Необходим большой объем оперативной памяти
- Желательно машина с кафедры СП

Доступ к данным



- Каким образом программа, запущенная на другой машине, может получить доступ к требуемым данным на моей машине?
  - Разделяемая файловая система
  - Передача файлов средствами Condor
    - Автоматическая отсылка обратно измененных файлов
    - Одновременная передача нескольких файлов
    - Шифрование пересылаемых данных
  - Remote I/O Socket
    - Требует подключения дополнительных библиотек
  - Механизм удаленных системных вызовов
    - Требует запуска в Standard Universe
    - Позволяет прозрачных образом передавать системные вызовы, генерируемые заданием, на машину пользователя, где хранятся необходимые данные
    - Позволяет пользователю запускать задания, не имея учетных записей на удаленных машинах

Отказоустойчивость

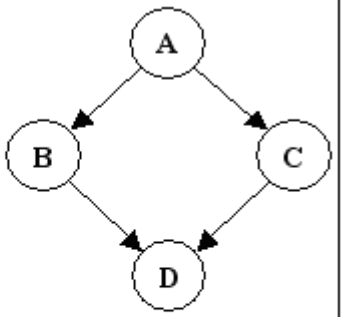


- Механизм *контрольных точек*
  - Доступен при запуске в Standard Universe
- **Контрольная точка** - полный набор информации, описывающий состояние выполнения задания (программы)
- Периодическая генерация контрольных точек во время выполнения задания
- Использование контрольных точек для
  - Возобновления выполнения задания после сбоя
  - Миграции задания на другую машину
- Отказы
  - При отказе машины, на которой выполняются вычисления, будет произведен откат и перенос выполняющихся заданий на другие машины
  - Отказ машины, отправившей задание, приведет к откату её заданий и продолжению их выполнения после её перезагрузки
  - Отказ центрального менеджера временно приведет к невозможности запуска новых заданий, но выполняющиеся задания не будут затронуты

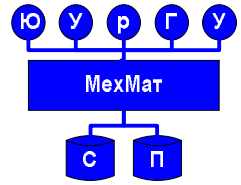
Механизм DAGMan

- Позволяет добавлять несколько заданий с зависимостями между ними
- Пример

```
# Filename: diamond.dag
#
Job  A  A.condor
Job  B  B.condor
Job  C  C.condor
Job  D  D.condor
Script PRE  A top_pre.csh
Script PRE  B mid_pre.perl  $JOB
Script POST B mid_post.perl $JOB $RETURN
Script PRE  C mid_pre.perl  $JOB
Script POST C mid_post.perl $JOB $RETURN
Script PRE  D bot_pre.csh
PARENT A CHILD B C
PARENT B C CHILD D
Retry  C 3
```



- Directed Acyclic Graph Manager (DAGMan)
  - **`condor_submit_dag diamond.dag`**

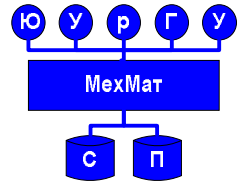


Фоновые процессы Condor

Функционирование системы поддерживается при помощи восьми фоновых процессов:

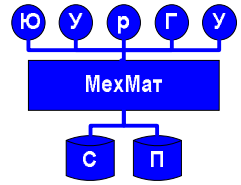
- condor_master
- condor_collector
- condor_negotiator
- condor_startd
- condor_starter
- condor_schedd
- condor_shadow
- condor_ckpt_server

condor_master



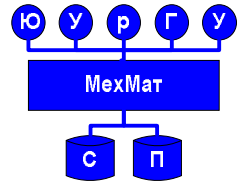
- контроль над всеми остальными фоновыми процессами
- запуск фоновых процессов
- повторный запуск неожиданно завершенных процессов и отсылка уведомления администратору
- завершение процессов по требованию администратора
- выполняется на всех узлах системы

condor_collector



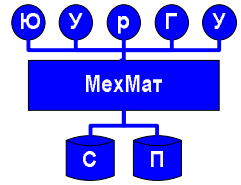
- выполняется на центральном менеджере
- собирает и хранит информацию о состоянии всех узлов системы
- все другие процессы периодически посылают ему данные о своем состоянии (ClassAds)
- обслуживает запросы от процессов и пользователей
- как минимум один процесс на систему

condor_negotiator



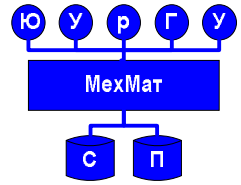
- Выполняется на центральном менеджере
- Осуществляет подбор ресурсов для заданий
  matchmaking - сопоставление требований заданий с доступными ресурсами
- Цикл подбора ресурсов (около 5 минут):
  - Получение информации от collector о доступных машинах и незапущенных заданиях
  - Сопоставление заданий и машин – оба описания должны удовлетворять требованиям друг друга
- Только один процесс на систему

condor_startd



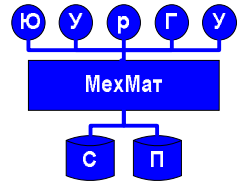
- запущен на всех выполняющих узлах
- представитель машины в системе
- отвечает за запуск и останов заданий пользователей
- следит за соблюдением политики доступа, установленной владельцем
- создает процесс condor_starter для каждого запущенного задания

condor_starter



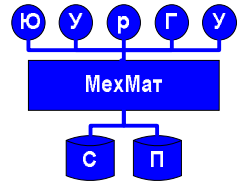
- порождается процессом condor_startd на выполняющей машине для каждого запущенного задания
- формирует среду для запуска заданий и мониторинга их выполнения
- пересылает системные вызовы от задания на запускающую машину
- после завершения задания посылает уведомление запускающему узлу и прекращает свою работу

condor_schedd



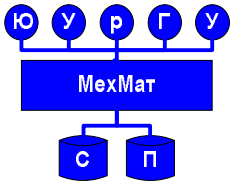
- выполняется на всех запускающих узлах системы
- управляет очередью запущенных заданий
- связывается с машинами о отправляет им задания
- позволяет пользователю управлять своими заданиями
- создает процесс condor_shadow для каждого запущенного задания

condor_shadow



- выполняется на запускающих узлах системы для каждого запущенного задания
- принимает пересылаемые системные вызовы с выполняющего узла, и отправляет их результаты обратно

Схема размещения фоновых процессов



= Process Spawned

= ClassAd Communication Pathway

Central Manager

Master

negotiator

schedd

Collector

Cluster Node

Master

startd

Cluster Node

Master

startd

Desktop

Master

startd

schedd

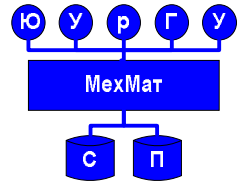
Desktop

Master

startd

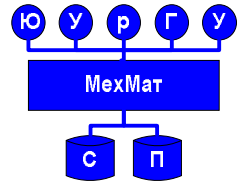
schedd

Condor и Globus Toolkit



- Виртуальный кластер Condor может быть интегрирован Grid на основе технологий Globus Toolkit
- => Компонент Condor-G позволяет использовать ресурсы Grid как часть виртуального кластера Condor
- <= Пользователи Grid могут получать доступ к виртуальному кластеру Condor средствами Globus Toolkit
- Симбиоз Condor и Globus Toolkit
  - Condor: значительно упрощает одновременное добавление и управление множеством заданий в Grid
  - Globus Toolkit: решения проблем, связанных с безопасностью, доступом к суперкомпьютерам и размещением исполняемого кода

Ссылки и литература

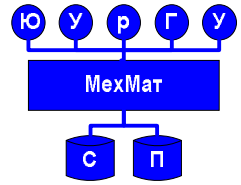


- http://www.cs.wisc.org/condor

- Chapter 11, Condor and the Grid, D. Thain, T. Tannenbaum, and M. Livny, Grid Computing: Making The Global Infrastructure a Reality, F. Berman, A. J. G. Hey, and G. Fox, editors, John Wiley, 2003.

- “Condor-G: A Computation Management Agent for Multi-Institutional Grids,” J. Frey, T. Tannenbaum, I. Foster, M. Livny, S. Tuecke, Proc. 10th Int. Symp. High Performance Distributed Computing (HPDC-10) Aug. 2001.

Презентации



9	7 ноября 2008 г.	Лекция 10. Системы разработки и поддержки P2P-приложений.	1. Платформа для распределенных вычислений BOINC. 2.Платформа для организации P2P сетей JXTA.
10	14 ноября 2008 г.	Лекция 11. Системы разработки и поддержки распределенных вычислительных систем.	1. Платформа для разработки распределенных сервисно-ориентированных систем WCF. 2. Объектно-ориентированная платформа для построения метасистем Legion
11	21 ноября 2008 г.	Лекция 12. Системы разработки и поддержки грид-сред.	1. Грид-система UNICORE. 2. Платформа для разработки и поддержки распределенных вычислений GPE

Спасибо за внимание!

Ваши вопросы?

Страница курса:

http://dom.susu.ru/grid.htm

Радченко Глеб Игоревич, каф. СП, ЮУрГУ