УДК 681.3

А. А. ПРИХОЖИЙ, О. М. ФРОЛОВ, Белорусский национальный технический университет

ИССЛЕДОВАНИЕ ПЛАНИРОВЩИКОВ ЗАДАЧ В GRID

Проблема управления выполнением заданий клиентов и оптимального использования распределенных вычислительных ресурсов является одной из ключевых в grid. В статье проведено детальное исследование и сопоставление планировщиков и принципов планирования решения задач в grid как по качеству работы, так и производительности. Результаты проведенных вычислительных экспериментов выявили критические компоненты планировщиков и приоритетные подходы к планированию, которые являются перспективными и определяют пути дальнейшего развития методов оптимизации grid-систем.

The problem of execution management of client's tasks and optimal management of cluster's computing resources is one of the key problems in grid. In this paper the detailed investigation and comparison of task schedulers and task scheduling techniques has been performed with respect to the result quality and throughput. The results of conducted computing experiments have found out the critical scheduler's components and preferable scheduling approaches which are promising and indicate the ways of further development of grid optimization methods.

Введение

Grid – это согласованная, открытая и стандартизованная среда, обеспечивающая гибкое, безопасное и скоординированное разделение вычислительных ресурсов в рамках одной виртуальной организации [1]. Grid-вычисления относятся к разновидности распределенных вычислений, где виртуальный суперкомпьютер представлен кластерами разнородных компьютеров, соединенными сетью и выполняющими огромное количество заданий, поступающих от разных клиентов. Ситуация неконтролируемого разделения времени большого числа процессоров посредством «ручного» управление заданиями, когда много параллельных задач конкурируют за вычислительные ресурсы, катастрофична для общей производительности системы. В связи с этим все современные вычислительные кластеры имеют систему управления ресурсами и планирования выполнения заданий, позволяющую оптимизировать использование ресурсов и обслуживание клиентов.

Решение задач в grid

Одним из важнейших компонентов grid-системы является GRAM (Grid Resource Al location and Management). Это набор сервисов, реализующих услуги по регистрации задач и ресурсов, формированию и обслуживанию очередей задач и ресурсов, отправки задач на выполнение на выделенных им ресурсах, мониторингу состояния, определению моментов завершения выполнения задач. Планирование решения задач на ресурсах *grid* является самостоятельной и чрезвычайно сложной проблемой как с точки зрения структурной сложности системы, так и с точки зрении вычислительных затрат на поиск оптимального плана решения [2].

Цель планирования – обеспечить скоординированное разделение ресурсов, учитывающее интересы пользователей и владельцев ресурсов. Результатом планирования является план использования ресурсов, учитывающий требования, ассоциированные с задачами. Представляют интерес методы и алгоритмы планирования, находящие наиболее оптимальное соответствие ресурсов задачам и устанавливающие точные временные интервалы времени, на которых ресурсы доступны задачам.

Вычислительные ресурсы *grid* обычно управляются локальным менеджером, который назначает ресурсы задачам в режиме конкуренции с другими задачами, находящимися в очереди на выполнение, в расчете на общую эффективность и высокую производительность. GRAM не является планировщиком ресурсов, он реализует протокол взаимодействия

с разнообразными локальными планировщиками посредством унифицированных сообщений.

Весь функционал, предоставляемый системами управления кластерами, можно разделить на две группы. Группа команд управления ресурсами обеспечивает: постановку задач в очередь на выполнение; планирование выполнения задач; запуск задач; остановку задач. Группа команд мониторинга обеспечивает: получение актуальной информации о текущем состоянии узлов вычислительной системы; получение информации о текущем состоянии задач. Центральным компонентом системы управления кластером является планировщик. Именно он обеспечивает эффективное использование вычислительных ресурсов узлов кластера. По мере поступления задач в очередь на выполнение, каждая получает некоторый приоритет. Планировщик выбирает наиболее приоритетные задачи и, в соответствии с некоторой стратегией планирования, выбирает предпочтительные узлы, на которых задачи исполняются.

Планировщики решения задач

Компонент GRAM grid-системы конструируется не в жесткой связке с конкретным планировщиком решения задач, а в качестве интегрирующего средства, допускающего встраивание разнообразных планировщиков, таких как Condor, PBS, SGE, Fork и др. Проанализируем сильные и слабые стороны каждого из них.

Планировщик *Condor* [3] разработан университетом Висконсин, США и является специализированной системой управления загрузкой распределенных ресурсов grid для задач большой вычислительной сложности. Он использует механизм очередей задач, приоритеты задач, стратегию планирования и мониторинга задач, а также стратегию управления распределенными ресурсами. В процессе функционирования Condor использует различные среды, например, Standard и Vanilla для выполнения последовательных задач, PVMи MPI для выполнения параллельных задач и др. Он допускает миграцию процессов путем приостановления их выполнения на одних машинах и продолжения выполнения на других незагруженных машинах. Приоритет задачи $\pi(u, t)$, сформированной пользователем u, меняется с течением времени tи рассчитывается по формуле

$$\pi(u,t) = \left[\beta\pi(u,t-\delta) + (1-\beta)\rho(u,t)\right] f(u,t), \quad (1)$$

где δ – интервал времени пересчета приоритетов; $\beta = 0.5^{\delta/h}$ – коэффициент скорости изменения приоритета; h – константа, заданная администратором; $\rho(u, t)$ – количество ядер используемого процессора; $f(u, t)$ – фактор поднятия приоритета пользователя uв момент времени t.

Планировщик *Sun Grid Engine* (*SGE*) [4] разработан SunMicrosystems (позже Oracle и Univa) с целью максимально эффективного использования ресурсов кластера компьютеров. Он выполняет прием, планирование, диспетчеризацию и управление удаленным выполнением большого числа одиночных, параллельных и взаимодействующих задач. SGE разбивает ресурсы на списки ожидания и назначает их задачам, находящимся в очереди на выполнение. Для каждого пользователя определяется квота общих доступных ресурсов. Благодаря алгоритму *backfill,* SGE ориентирован на эффективное планирование большого потока малых задач.

Планировщик *Portable Batch System* (*PBS*) [5] впервые разработан NASA с целью распределения вычислительных задач между доступными ресурсами кластера. Он принимает сценарий оболочки, пакетные задания и управляющие атрибуты, сохраняет и защищает задания до запуска, запускает их на выполнение и передает результаты клиенту. Планировщик *Torque* относится к семейству PBS и является менеджером ресурсов Linux-кластера, использующим средства управления параллельными процессами, такие как MPI. Он обеспечивает запуск процессов, передачу данных, сборку мусора, создание интерактивных сессий, разделение процессорного времени между задачами и др. PBS включает четыре главных компонента: команды, сервер заданий, планировщик заданий и исполнитель заданий. Современная версия *Torque* использует развитый планировщик *Maui*, который может выполнять большое количество задач и выравнивать загрузку узлов.

Планировщик *Fork* [6] запускает разветвляющиеся задачи на выполнение и отслеживает изменение статуса выполняемых задач

посредством генераторов событий. Стартер разветвляющихся задач создается для каждой родительской задачи и завершает выполнение при завершении каждой из подчиненных задач. Планировщик *Fork* не является кластерным и не реализует мониторинг выполняемых задач. Планировщик реализует стратегию раньше пришел – раньше обслуживается (First Come First Served – FCFS).

Проблема малых заданий, ассоциирующаяся с планированием многопроцессорных заданий, является наиболее трудной для планировщика. Поскольку много малых заданий фрагментируют ресурсы, приоритетное планирование не способно обеспечить запуск заданий, требующих объемных ресурсов. Алгоритмы обратного заполнения *backfill* и справедливого распределения ресурсов *fairshare* [7] способны накапливать необходимые ресурсы и представляются лучшими известными алгоритмами планирования многопроцессорных заданий в grid.

Организация экспериментов с планировщиками

Эксперименты проводились на тестовом вычислительном кластере, архитектура которого показана на рис. 1. Развертывание кластера выполнено посредством веб-сервиса Amazon Cloud и инструментария Globus Toolkit. Компоненты службы безопасности Globus-gsi, приема задач и управления ресурсами Globus-Gram и хранения/передачи файлов Globus-gridftp назначены на отдельные узлы кластера, аппаратное обеспечение каждого из которых включает процессор Intel Xeon E54302.66 GHz и оперативную память емкостью 600 МБ с доступом к сети со скоростью 23.2 MB/s. На узле Globus-Gram установлена мастер-часть планировщика, отвечающая за прием и рассылку задач на свободные вычислительные узлы, а также адаптер между Globus-Gram и планировщиком. Выполнение планируемых задач осуществлялось на двух вычислительных узлах, обладающих 4-х ядерными процессорами Xeon E5-2680 v2 (Ivy Bridge) Processors 2.8 GHz и оперативной памятью емкостью 7.5 ГБ с доступом к сети со скоростью 23.2 MB/s. На этих узлах установлена часть планировщика, принимающая задачи от Gram и запускающая их на выполнение.

Клиент в течение часа случайным образом генерирует задачи на вычисление чиселπ с различной точностью. Число $n(\pi)$ значащих цифр решающим образом влияет на потребляемые задачей вычислительные ресурсы: объем памяти и интервал процессорного времени. Число $n(\pi)$ выбирается из конечного множества по равномерному закону распределения вероятностей. В начале теста задачи генерируются непрерывно до заполнения очереди из 50 задач. Далее клиент ожидает результаты выпол-

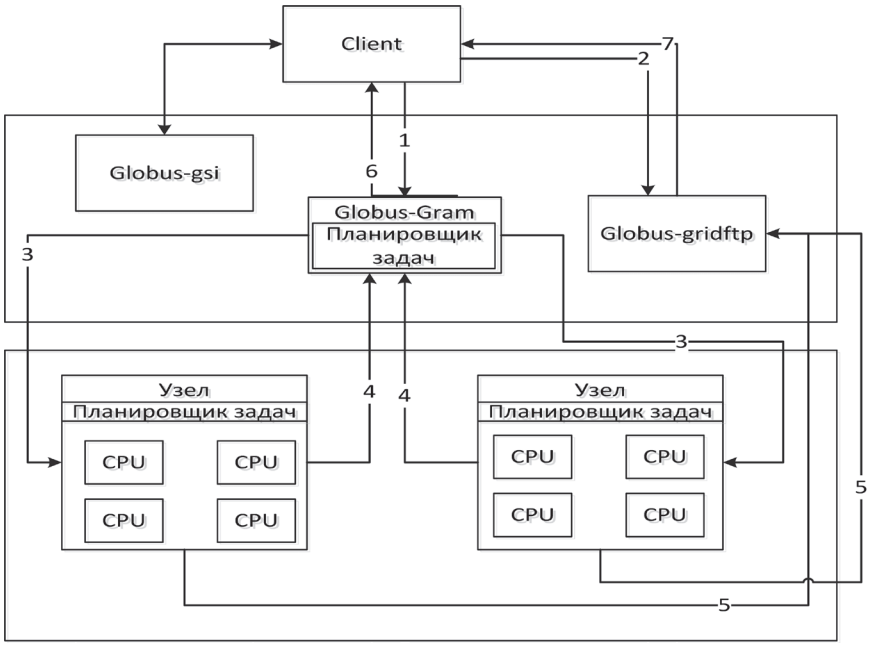

Рис 1. Архитектура вычислительного кластера

нения задач. Получение результата от очередной выполненной задачи влечет генерацию и постановку в очередь следующей задачи. По истечении часа клиент ждет завершения всех ранее сгенерированных задач.

В каждой задаче число π вычисляется алгоритмом Чудновского [8], в основе которого лежит уравнение

$$\frac{1}{\pi} = 12\sum_{k=0}^{\infty} \frac{(-1)^k (6k)!(545140134k + 13591409)}{(3k)!(k!)^3 640320^{3k+3/2}}, \quad (3)$$

где k – член ряда. В зависимости от $n(\pi)$ задачи имеют различные требования к ресурсам памяти и процессорного времени (табл. 1). При увеличении числа значащих цифр в π с 10000 до 30000000 требуемый объем памяти возрастает в 420 раз, а требуемое процессорное время – в 3504 раз на одном процессоре.

Таблица 1. **Объем памяти и процессорного времени, требуемый алгоритмом Чудновского в зависимости от числа значащих цифр в числе π**

Количество цифр $n(\pi)$	Требуемый объем памяти (MB)	Требуемое время на одном процессоре (с)
10000	1	0.007
100000	10	0.041
1000000	80	0.511
10000000	136	7.121
20000000	291	14.967
30000000	420	24.530

После генерации каждая задача поступает в компонент Globus-Gram. Над ней выполняются следующие операции: 1) постановка задачи в очередь на планирование; 2) запись тестовых данных на узел Globus-gridftp; 3) планирование выполнения задачи; 4) запуск задачи на узле выполнения; 5) формирование результата выполнении; 6) запись результата на узел Globus-gridftp; 7) получение клиентом информации о завершении задачи; 8) передача клиенту результирующих данных с Globus-gridftp. Описанный сценарий был последовательно реализован для планировщиковCondor, PBS, SGE иFork.

Результаты вычислительных экспериментов

Табл. 2 описывает задачи, сгенерированные и выполненные в экспериментах с каждым из планировщиков. Например, в эксперименте с планировщиком SEG сгенерировано 262 задачи с $n(\pi)$ = 10000, из которых 134 задачи выполнены на узле *1* и 128 задач выполнены на узле *2*. Общее число сгенерированных задач является случайным, но зависит от производительности и качества работы планировщика. Предпоследняя строка табл. 2 описывает суммарное время, прогнозируемое в соответствии с табл. 1 на выполнение всех задач с одинаковым $n(\pi)$ на одном процессоре. Так, для задач с $n(\pi)$ = 10000 прогнозируемое время равно 6.8 с, а для задач с $n(\pi)$ = 30000000 время равно 24064 с. Соответственно, доля первых задач в общем прогнозируемом времени составляет только 0.015%, а доля вторых задач –51.59% (последняя строка табл. 2). Видим, что трудоемкость задач варьируется в широком диапазоне, что создает среду исследования планировщиков в ситуациях с большой фрагментацией ресурсов. Прогнозируемое время для каждого из планировщиков на каждом из узлов *1* и *2* описывается предпоследним столбцом табл. 2. Максимальная разница прогнозируемой загруженности узлов составила

Таблица 2. **Задачи, сгенерированные клиентом для каждого планировщика и каждого узла**

Планировщик	Узел	Количество цифр n(π) в числе π						Время (с)	Время (с)
		10000	100000	1000000	10000000	20000000	30000000		
Condor	1	133	120	119	128	110	113	5396	10843
	2	118	132	110	122	105	120	5446	
PBS	1	106	110	115	127	124	101	5302	11374
	2	123	117	115	135	111	138	6072	
SEG	1	134	161	141	123	161	141	6824	13433
	2	128	138	130	136	136	144	6609	
Fork	1	129	126	129	126	122	103	5322	10994
	2	97	123	91	116	122	121	5672	
Суммарное время (с)		6.8	42	485	7213	14832	24064	46644	46644
Доля во времени (%)		0.015	0.09	1.04	15.47	31.80	51.59	100	100

Таблица 3. **Производительность планировщиков**

Планировщик	Время выполнения всех задач (с)	Ускорение (раз)	Количество выполненных задач			Среднее потребление планировщиком времени процессора (%)		
			узел 1	узел 2	всего	узел 1	узел 2	мастер узел
Condor	3867	2.80	723	707	1430	2.06	2.05	3.6
PBS	3910	2.91	683	739	1422	3.96	4.06	2.6
SGE	3825	3.51	861	812	1673	1,45	1.43	4.5
Fork	4107	2.68	735	670	1405	2.4	2.4	–

12.7% для планировщика PBS, средняя разница по всем планировщикам – 5.8%. В разнице загруженности узлов проявляется качество планирования многопоточных задач.

Важнейшие параметры, характеризующие производительность планировщиков, приведены в табл. 3. Качество работы планировщика оценивается тремя параметрами: общим временем выполнения всех задач, ускорением параллельного выполнения задач по сравнению с последовательным выполнением, количеством спланированных и выполненных задач за все время работы планировщика. Общее время планирования и выполнения задач складывается из двух частей: времени генерации и приема задач на выполнение, равного 3600 с, и времени завершения задач, оставшихся по истечении 1 ч. Так для планировщиков Condor, PBS, SGE и Fork время завершения составило 267, 310, 225 и 507 с соответственно. Ускорение параллельного выполнения многопоточных задач по сравнению с последовательным выполнением эквивалентных однопоточных задач составило 3.51, 2.91, 2.8 и 2.68 для планировщиков SGE, PBS, Condor и Fork соответственно. Наиболее быстрый планировщик SGE обеспечил выполнение максимального числа задач 1673. Планировщики Condor, PBS и Fork обеспечили выполнение 1430, 1422 и 1405 задач соответственно, что на 14.5%, 15.0%, и 16.0% меньше по сравнению с SGE.

Быстродействие планировщика оценивается долей процессорного времени, потребляемого в процессе функционирования. Согласно табл. 3, столбцы 7 и 8, потребление планировщиками времени на вычислительных узлах *1* и *2* составило: SGE – 1.44%, Fork – 2.40%, Condor – 3.69% и PBS – 4.01%. Потребление времени на мастер-узле составило: Fork – 0%, PBS – 2.6%, Condor – 4.1% и SGE – 4.5%. Более детальный анализ процессорного времени, потребляемого планировщиками, дают графики, изображенные на рис. 2.

Вид графиков потребления процессорного времени кластерным планировщиком Condor определяется используемой стратегией планирования – многоуровневой очередью с приоритетами, рассчитываемыми динамически по характеристикам пользовательских задачах. Пики потребления процессорного времени периодически повторяются каждые 10 с и совпадают с моментами пересчета приоритетов задач. Стадия инициализации внутренних подсистем Condor длительностью около 4 с определяет пик потребляемого процессорного времени до 25% на вычислительном узле и 55% на мастер-узле.

Вид графика потребления времени кластерным планировщиком PBS определяется стратегией Backfill, стремящейся к справедливому распределению ресурсов grid. Высокоприоритетным многопоточным задачам предоставляется возможность параллельного выполнения сразу на нескольких процессорах. Планировщик способен накапливать и резервировать для трудоемких задач необходимый объем фрагментированных ресурсов. Стадия инициализации внутренних подсистем продолжается 12 с с потреблением около 15% процессорного времени на вычислительном узле и 40% времени на мастер-узле.

Кластерный планировщик SGE дает лучшее качество результатов планирования, но потребляемое им процессорное время превышает время, потребляемое планировщиком Fork. Не смотря на то, что SGE используют ту же стратегию планирования, что и PBS, существенное различие в объеме потребляемого времени (1.44% против 4.01%) свидетельствует о разных механизмах обработки очереди задач. Стадия инициализации внутренних подсистем SGE-Gramсамая длинная (16 с). Собственное потребление вычислительных ресурсов планировщиком SGE сокращается при отключении мониторинга запущенных на выполнение задач.

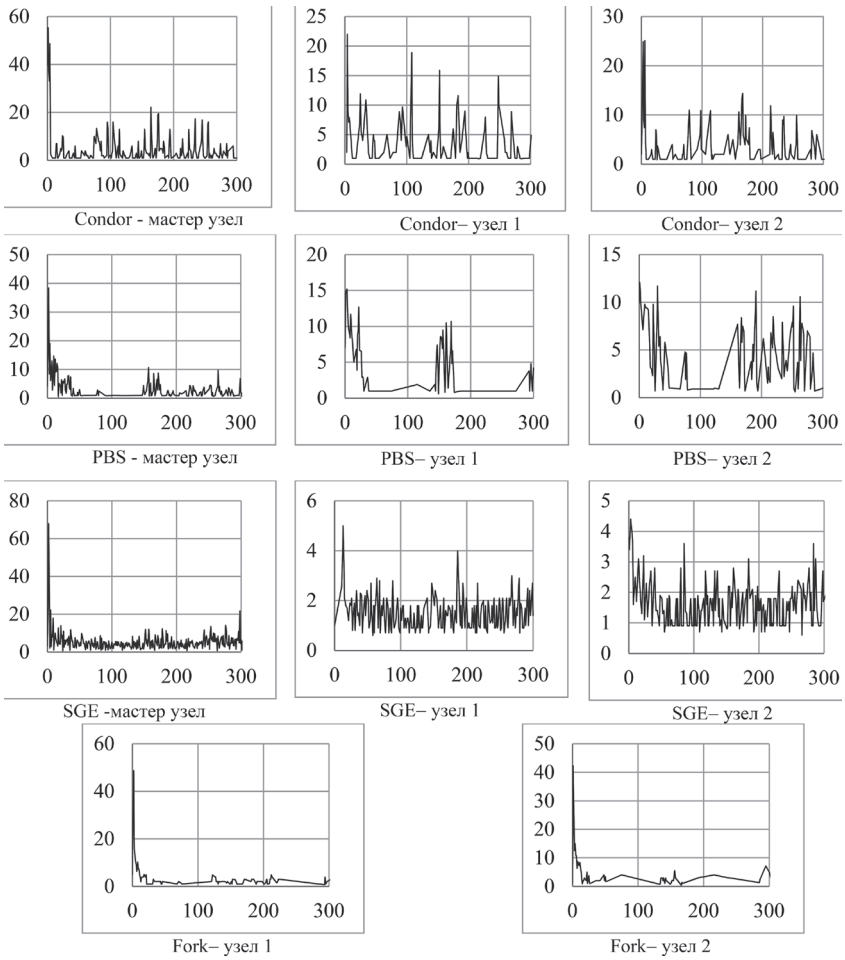

Рис. 2. Потребление планировщиками процессорного времени в процентах первые 300 с

Анализ графиков загрузки вычислительных узлов планировщиком Fork показывает, что существует высокий пик (до 50%) периода инициализации. Достаточно низкое потребление времени в стационарном режиме объясняется тем, что планировщик не является кластерным, не присутствует на мастер-узле, реализует достаточно простую стратегию планирования и не осуществляет мониторинг задач в период выполнения.

Среднее, средневзвешенное и пиковое потребление процессорного времени процессами сервисаGram и процессами каждого из четырех планировщиков приведено в табл. 4. Средневзвешенное потребление оценивается лишь на интервалах времени, на которых наблюдается отличная от нуля активность процесса.

Для поддержки функционирования всех планировщиков используются два процесса службы Gram: процесс globus-gatekeeper авторизации и поддержки работы пользователя; процесс globus-job-manager управления планированием, выполнением и отслеживанием состояния задач. Процесс globus-scheduler обработки событий относится только к планировщикам PBS и SGE, использующим SEG-адаптер (Scheduler Event Generator) для Gram. Активность процессов убывает в порядке globus-job-manager, globus-gatekeeper, globus-scheduler. Высокое потребление времени этими процессами ставит задачу оптимизации работы SEG-адаптера, обеспечивающего взаимодействие Gram и планировщика.

В планировщике Condor наибольшим потреблением среднего, средневзвешенного и пикового процессорного времени обладают процесс condor_procd отслеживания и управления выполнением задачи, процесс condor_schedd планирования задач из очереди ожидания и процесс condor_startd взаимодействия между вычислительным узлом и мастер-узлом Condor. Средневзвешенное и пиковое потребление времени выстраивает оставшиеся процессы

Таблица 4. **Потребление временипроцессами Gram и процессами планировщиков**

Процессы планировщиков	Среднее потребление (%)		Средневзвешенное потребление (%)		Пиковое потребление (%)	
Condor						
	мастер узел		мастер узел		мастер узел	
globus-job-manager	1.3677		9.3365		37.6	
condor_schedd	0.9427		3.1493		14.5	
condor_startd	0.8413		2.1568		6	
condor_shadow	0.4513		2.0718		6.9	
condor_collector	0.2365		1.1236		3	
condor_negotiator	0.1647		1.4106		3	
condor_starter	0.0590		1.8322		5	
globus-gatekeeper	0.0416		1.0807		2	
condor_master	0.0037		1.2000		2	
	узел1	узел2	узел1	узел2	узел1	Узел2
condor_procd	1.5911	1.6815	4.5907	4.8567	12.9	15.6
condor_schedd	1.0846	1.0803	2.6184	2.7284	12.4	14.5
condor_startd	0.9697	0.9623	1.8358	1.8880	5.9	5
condor_master	0.0042	0.0043	1.0000	1.0000	1	1
PBS						
	мастер узел		мастер узел		мастер узел	
globus-job-manager	1.4458		3.5556		28.6	
pbs_server	0.6261		2.3424		35.3	
pbs_schedd	0.2607		1.0280		2	
globus-gatekeeper	0.2388		3.7308		15.2	
globus-scheduler	0.0535		3.4785		13.8	
	узел1	узел2	узел1	узел2	узел1	узел2
pbs_mom	3.7224	3.8253	5.9256	5.7199	14	13.5
pbs_shed	0.2497	0.2375	0.9611	0.9102	1.9	1.9
SGE						
	мастер узел		мастер узел		мастер узел	
globus-job-manager	2.4126		4.6111		57.6	
sge_qmaster	1.8355		4.7607		68.4	
globus-gatekeeper	0.1562		2.9858		10.6	
globus-scheduler	0.0602		7.7013		31	
sge_shepherd	0.0271		0.9874		2.7	
	узел1	узел2	узел1	узел2	узел1	узел2
sge_execd	1.4588	1.4335	5.7284	3.3169	20.2	20.3
Fork						
	узел1	узел2	узел1	узел2	узел1	узел2
globus-job-manager	2.42	2.41	3.77	2.05	48.9	50.3
globus-gatekeeper	0.07	0.08	0.97	0.96	1	1

в следующем порядке: condor_shadow – процесс сопровождения выполняемой задачи; condor_collector – процесс сбора данных с других служебных процессов; condor_negotiator – процесс сопоставления задач с доступными ресурсами; condor_starter – процесс обработки всех запросов, поступающих от выполняемой задачи; condor_master–ведущий процесс планировщика.

В планировщике PBS наибольшим потреблением среднего, средневзвешенного и пикового процессорного времени обладает процесс pbs_mom запуска мини сервера для выполнения групп задач. Процесс pbs_server реализации сервера PBS имеет большее средневзвешенное и пиковое потребление по сравнению с процессом pbs_sched планирования задач.

В планировщике SGE существенно наибольшим потреблением среднего, средневзвешенного и пикового процессорного времени обладают процессы sge_qmaster службы управления и sge_execd агента выполнения сплани-

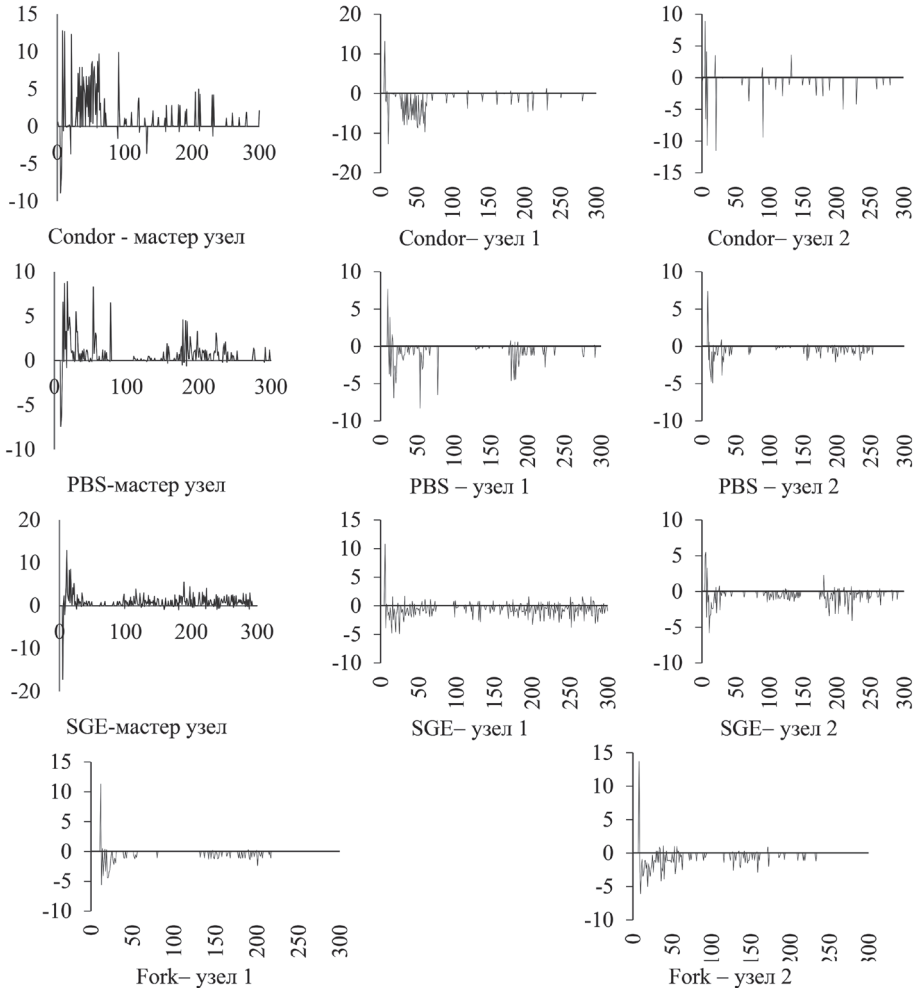

Рис. 3. Графики входящего и исходящего сетевого трафика

рованных задач. Процесс sge_shepherd агента контроля состояния выполняемых задач имеет значительно меньшее потребление.

Планировщик Fork реализуется только процессами globus-gatekeeper и globus-job-manager службы Gram, причем пиковое потребление времени процессом globus-job-manager в Fork значительно превосходит его же потребление в Condor и PBS.

Важнейшим параметром, характеризующим качество работы каждого из планировщиков, является нагрузка на сеть (рис. 3), измеряемая входящим (положительная область оси ординат) и исходящим (отрицательная область оси ординат) трафиками (время указано в секундах). Планировщик Fork, не являющийся кластерным, обеспечил наименьший средний трафик по двум вычислительным узлам (входящий – 2.91 KB/s, исходящий – 3.21KB/s). Планировщик PBS сгенерировал наибольший трафик (вычислительный узел: входящий – 26.32 KB/s, исходящий –26.52 KB/s; мастер узел: входящий – 53.33 KB/s, исходящий – 53.33 KB/s). Если от входящего/исходящего на мастер-узел трафика отнять трафик на вычислительные узлы, то получим сетевую активность между мастер-узлом и клиентом. Трафик между клиентом и мастер-узлом у PBS составил: входящий – 0.29 KB/s, исходящий – 0.68 KB/s. Планировщик Condor сгенерировал средний трафик (вычислительный узел: входящий – 14.04 KB/s, исходящий – 14.26 KB/s; мастер-узел: входящий – 28.20 KB/s, исходящий – 27.20 KB/s). Трафик между клиентом и мастер-узлом у Condor составил: входящий – 0.31 KB/s, исходящий – 0.89 KB/s. Трафик, сгенерированный планировщиком SGE, хотя и превышает трафик, сгенерированный планировщиком Fork, однако с учетом богатой функциональности является достаточно низким (вычислительный узел: входящий – 6.60 KB/s, исходящий – 6.85 KB/s; мастер-узел: входя-

щий – 14.47 KB/s, исходящий – 13.95 KB/s). Трафик между клиентом и мастер-узлому SGE составил: входящий – 0.25 KB/s, исходящий – 1.28 KB/s. По возрастанию разности исходящего и входящего трафиков планировщики выстроились в следующем порядке: PBS – 0.192 KB/s, Condor – 0.213 KB/s, SGE – 0.251 KB/s, Fork – 0.3 KB/s. Анализируя вариации входящего трафика между клиентом и мастер-узлом для разных планировщиков заметим, что они незначительны и в значительной степени обусловлены количеством задач, отправленных на выполнение. Вариации в исходящем трафике сильно зависят от планировщика и связаны с объемом информации о процессе выполнения задач, передаваемой клиенту.

Заключение

Проведено исследование качества работы и производительности планировщиков выполнения задач с целью выявления критических компонентов и путей оптимизации grid-систем. Достоверность полученных результатов обусловлена одинаковыми условиями генерации и выполнения тестовых задач, а наблюдаемые различия в параметрах показывают особенности организации, алгоритмов работы и внутренней реализации планировщиков. Исследуемым и объектами явились менеджер ресурсов grid-системы, планировщик задач и адаптер между менеджером ресурсов и планировщиком задач. Исследована внутренняя структура каждого из четырех планировщиков и выявлены компоненты, потребляющие наибольший объем процессорного времени и являющиеся перспективными для оптимизации и улучшения параметров планировщиков.

Наилучшее качество результатов планирования дал планировщик SGE. Он обеспечил выполнение наибольшего числа задач за отведенный интервал времени, наилучшим образом справился с планированием параллельных многопоточных задач и дал коэффициент ускорения 3.51 для четырех потоков, выполняемых на четырех ядрах. Планировщик SGE ставит задачи в очередь и выполняет их планирование асинхронно, в силу этого график выполнения задач в отличие от планировщика PBS выглядит сильно неравномерным. Анализ сетевой активности при планировании и выполнении задач показывает схожее потребление сетевых ресурсов планировщиками SGE и PBS. Сказанное относится также к потреблению процессорного времени, поскольку оба планировщика реализуют один алгоритм планирования Backfill. В отличие от них Condor реализует алгоритм управления очередью по приоритетам задач, что увеличивает его сетевую активность, но уменьшает потребление процессорного времени.

Литература

1. **Foster I.** Computational Grids / I. Foster, C. Kesselman // Chapter in book «The Grid: Blueprint for a New Computing Infrastructure», Morgan-Kaufman, 1999.

2. **Xhafa F. and Abraham A.** Meta-heuristics for Grid Scheduling Problems // Springer-Verlag, Berlin-Heidelberg, SCI146, 2008. – P. 1–37.

3. **Thain D.** Distributed Computing in Practice: The Condor Experience / D. Thain, T. Tannenbaumand M. Livny // Concurrency and Computation: Practice and Experience, Vol. 17, No. 2–4, 2005. – P. 323–356.

4. **Gentzsch W.** Sun Grid Engine: Towards Creating a Compute Power Grid / W. Gentzsch // CCGRID, IEEE Computer Society, 2001. – P. 35–39.

5. **Henderson R.** Portable batch system: External reference specification / R. Henderson and D. Tweten // Technical report, NASA, Ames Research Center, 1996.

6. **Towsley D.** Analysis of Fork-Join Program Response Times on Multiprocessors / D. Towsley, C. G. Rommel, J. A. Stankovich // IEEE Trans. Parallel and Distributed Systems, Vol. 1, No.3, 1990. – P. 286–303.

7. **Gibbons R.** A historical application proler for use by parallel schedulers. In Job Scheduling Strategies for Parallel Processing, D. G. Feitelson and L. Rudolph (eds.), pp. 58{77, Springer Verlag, 1997. Lect. Notes Comput. Sci. Vol. 1291.

8. **Chudnovsky D. V.** The computation of classical constants / D. V. Chudnovsky, G. V. Chudnovsky // Proc. Nat. Acad. Sci. U. S. A. Vol. 86, No. 21, 1989. – P. 8178–8182.