Применение механизмов контроля насыщения (Congestion Control) для разделения ресурсов на распределенной вычислительной среде

Д. А. Сериков

НИИ Механики МГУ

3 февраля 2010 г.

Ключевые понятия

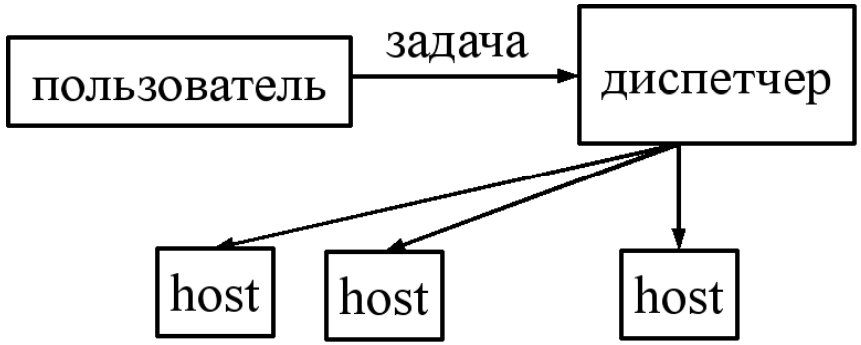
- *Ресурс* — логическая или физическая часть распределенной вычислительной среды Grid, которая может быть выделена пользователю.
- *Задача* — программная единица для обработки в распределенной вычислительной среде Grid.
- *Диспетчеризация задач* — автоматическая обработка набора задач, включающая в себя *планирование ресурсов*, доставку необходимых входных файлов на ресурс, запуск, управление и мониторинг выполнения задач, доставку выходных файлов.
- *Планирование ресурсов* — распределение задач по доступным ресурсам.

План доклада

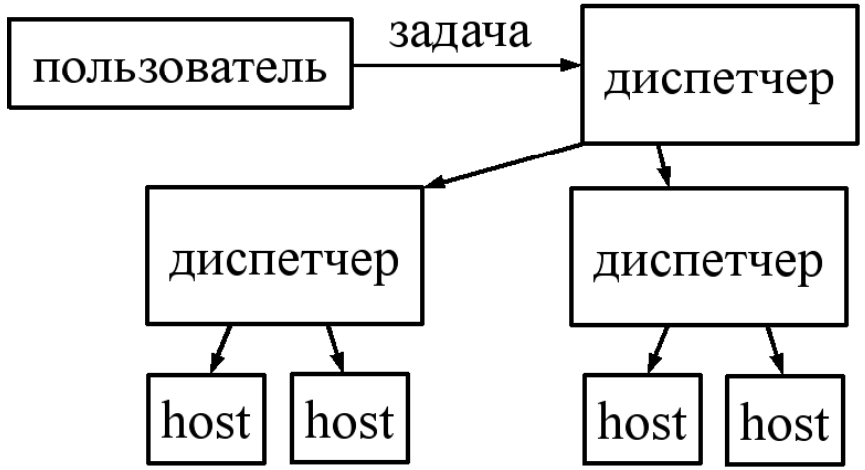
- Подходы к диспетчеризации задач в Grid
- Математическое моделирование диспетчеризации
  - Экспериментальное моделирование
  - Имитационное моделирование
- Алгоритм планирования с Congestion Control
  - Алгоритмы Slow Start и Congestion Avoidance
  - Результаты тестирования алгоритма
  - Имитационное моделирование алгоритма

Подходы к диспетчеризации задач в Grid

- Централизованная диспетчеризация задач

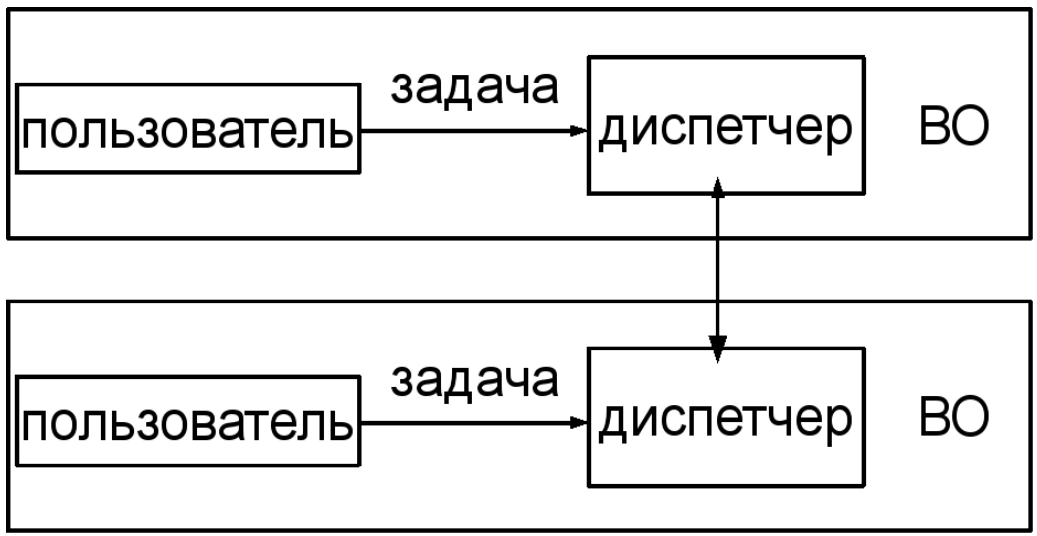


- Иерархическая диспетчеризация задач



Подходы к диспетчеризации задач в Grid

- Равноправная (децентрализованная) диспетчеризация задач



Экспериментальное моделирование

- Входные данные — параметры диспетчера GridWay.
- Выходные данные — значение функции утилизации ресурса.
- В качестве вычислительного приложения использовалась задача доказательства ненадежности криптоалгоритма RC5 путем взлома зашифрованного им сообщения.

Определения

Определение

Вычислительный ресурс — логическая или физическая часть распределенной вычислительной среды Grid, которая соответствует одной вычислительной машине или кластеру. Вычислительный ресурс может включать в себя один или несколько процессоров.

Определение

Функция утилизации процессора p_j на временном интервале $(t_1; t_2)$ определяется как отношение количества тактов процессора, потраченных на пользовательский режим на временном интервале $(t_1; t_2)$, к общему числу тактов процессора на этом интервале

$$f_j(t_1, t_2) = \frac{U_j(t_2) - U_j(t_1)}{T_j(t_2) - T_j(t_1)}.$$

Определения

Определение

Загрузка процессора p_j в момент времени t есть значение функции утилизации процессора p_j на временном интервале $(t - \Delta t; t)$

$$c_j(t) = c_j(t, \Delta t) = \frac{U_j(t) - U_j(t - \Delta t)}{k_j \Delta t},$$

где Δt — фиксированный отрезок времени.

Предположим, что $t_2 - t_1 = n\Delta t$, где n — целое, тогда функция утилизации для вычислительного ресурса h_i на временном интервале $(t_1; t_2)$ рассчитывается по формуле:

$$F_i(t_1, t_2) = \frac{\Delta t}{n_i(t_2 - t_1)} \sum_{p_j \in h_i} \sum_{l=0}^{n-1} c_j(t_2 - l\Delta t)$$

Параметры GridWay

- *SCHEDULING_INTERVAL* (*SI*) — период между двумя итерациями планирования;
- *DISCOVERY_INTERVAL* (*DI*) — период поиска новых хостов в Grid-полигоне;
- *MONITORING_INTERVAL* (*MI*) — период обновления сведений по каждому ресурсу;
- *POLL_INTERVAL* (*PI*) — период опроса о состоянии задач;
- *DISPATCH_CHUNK* (*DC*) — максимальное число задач, которые будут обработаны планировщиком за один шаг планирования;

Вычисление функции утилизации

Параметры GridWay					Утилизация ресурса, %			
SI	*DI*	*MI*	*PI*	*DC*	tp(2)	gf(2)	imec(16)	sscc(81)
30	900	300	180	15	62.58	75.76	66.31	51.36
15	900	300	180	15	65.82	81.00	75.99	53.38
60	900	300	180	15	53.98	69.88	63.75	51.31
30	450	300	180	15	66.70	80.36	61.18	45.51
30	1800	300	180	15	62.50	70.80	74.22	53.46
30	900	150	180	15	65.66	72.93	62.87	53.17
30	900	600	180	15	53.75	81.90	68.07	49.98
30	900	300	90	15	63.61	82.75	67.00	61.34
30	900	300	360	15	56.91	75.42	60.10	42.15
30	900	300	180	5	41.53	48.39	45.23	35.49
30	900	300	180	30	69.33	79.34	68.36	53.24

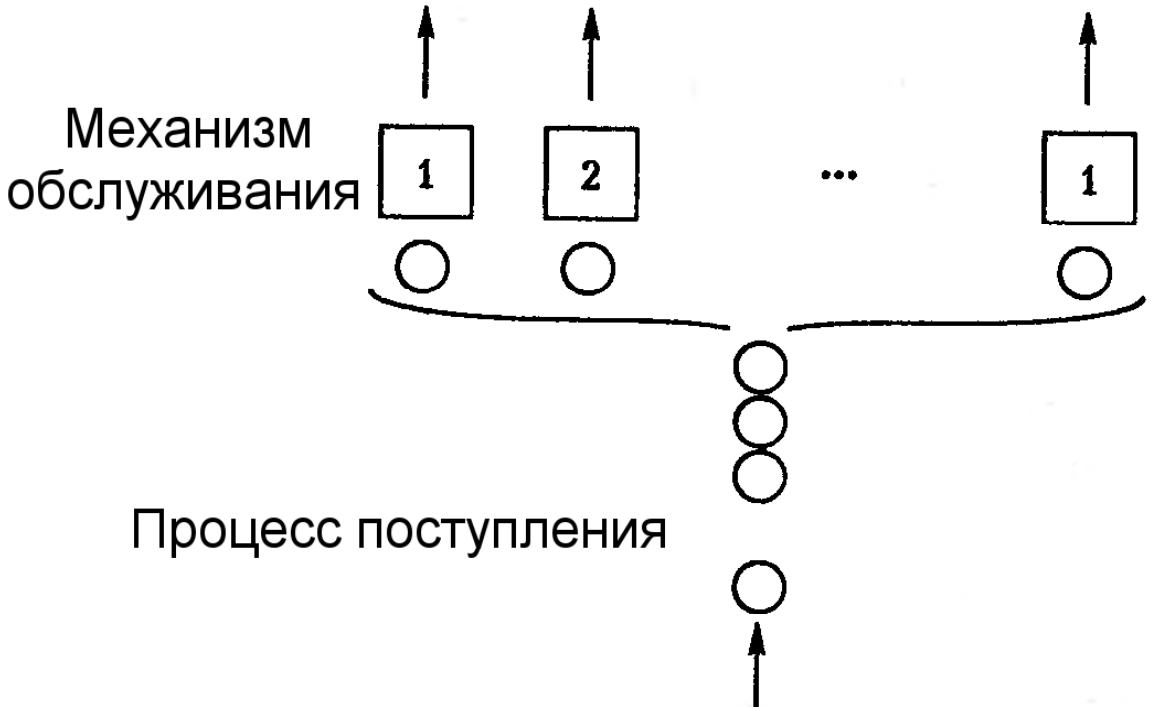
Grid-полигон

В качестве инфраструктурной основы для исследований функциональных возможностей и эффективности механизмов планирования ресурсов для решения вычислительных задач на Grid-среде использовался экспериментальный Grid-полигон, развернутый на базе высокопроизводительных вычислительных систем:

- Института механики МГУ (Москва);
- Научно-образовательного центра компьютерного моделирования и безопасных технологий (НОЦ КМиБТ, Москва),
- Института вычислительной математики и математической геофизики Сибирского отделения РАН (ИВМиМГ СО РАН, Новосибирск)

Моделирование систем массового обслуживания (Имитационное моделирование)

- Процесс поступления
- Механизм обслуживания
- Дисциплина обслуживания
  - Требования *FIFO*
  - Требования *LIFO*
  - Требования с *приоритетом*



Дискретно-событийная модель системы диспетчеризации задач, используемой в GridWay

1. Если хотя бы один из ресурсов свободен, то выборка задач из очереди и их обслуживание начинается не немедленно, а в моменты времени, кратные параметру SI, где $SI \geqslant 0$ — параметр системы.
2. Величины $N_0, N_1, N_2, \ldots$ — независимые и одинаково распределенные, представляющие общее количество ресурсов системы в моменты времени $0, DI, 2DI, \ldots$, где $DI \geqslant 0$ — параметр системы.
3. Величины $M_0, M_1, M_2, \ldots$ — независимые и одинаково распределенные, представляющие количество ресурсов системы, занятых локальными очередями соответственно в моменты времени $0, MI, 2MI, \ldots$, где $MI \geqslant 0$ — параметр системы.

Дискретно-событийная модель системы диспетчеризации задач, используемой в GridWay

4. Если обслуживание задачи i началось в момент времени kSI (пункт 1), то соответствующий ресурс освободится в момент времени

$$PI \left\lceil \frac{kSI + S_i}{PI} \right\rceil,$$

где $PI \geqslant 0$ — параметр системы.

5. За один шаг планирования, соответствующий выборке задач из очереди, из нее извлекается не более DC задач, где $DC > 0$ — параметр системы.

Система массового обслуживания GI/G/D/M(SI,DI,MI,PI,DC)

Определение

Система с перечисленными выше свойствами 1–5 называется системой массового обслуживания GI/G/D/M(SI,DI,MI,PI,DC). При этом GI относится к распределению величин A_i, G — к распределению величин S_i, D и M — к распределению величин N_i и M_i, соответственно.

При $N_0 = N_1 = N_2 = ... = N, M_0 = M_1 = M_2 = ... = 0$ система массового обслуживания $GI/G/N$ является предельным случаем системы $GI/G/D/M(SI,DI,MI,PI,DC)$ при $SI, PI \to 0, DC \to \infty$.

Критерии оценки работы системы диспетчеризации задач, используемой в GridWay

- D_i — время задержки в очереди задачи i
- $Q(t)$ — число задач в очереди в момент времени t
- $B(t)$ — число занятых ресурсов

$$d = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} D_i$$

— *установившаяся средняя задержка*

$$Q = \lim_{T\to\infty} \frac{1}{T} \int_0^T Q(t)\,dt$$

— *установившееся среднее по времени число задач в очереди*

$$u = \lim_{T\to\infty} \frac{1}{T} \int_0^T B(t)\,dt$$

— *установившийся коэффициент использования ресурсов*

Результаты компьютерного моделирования

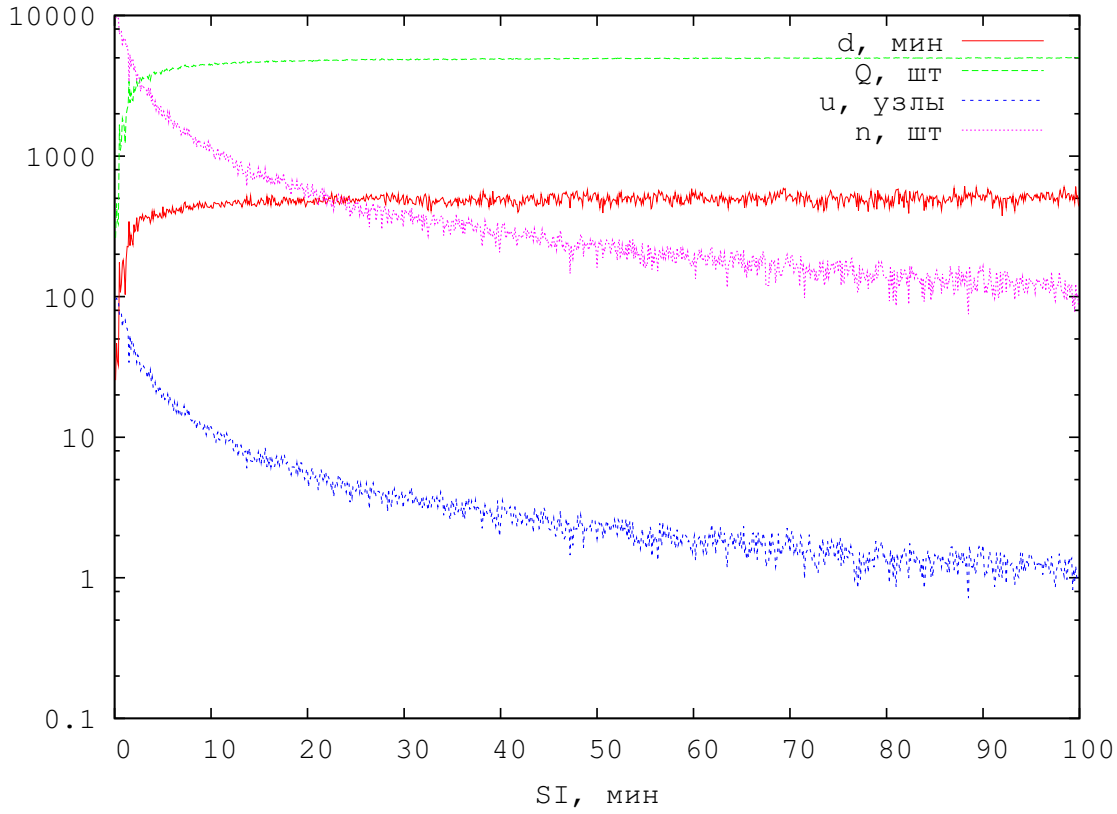


Рис.: Зависимость показателей эффективности от периода времени между двумя итерациями планирования

Предпосылка алгоритма

- Метадиспетчер msh для иерархической диспетчеризации
- Планирование в среде с «ошибочной» информацией о ресурсах

Алгоритмы Slow Start и Congestion Avoidance (RFC 2581)

- Переменные алгоритмов
  - размер окна насыщения $cwnd = cwnd_i$;
  - анонсируемое узлом окно $rwnd$;
  - число активных задач $FlightSize$;
  - порог насыщения $ssthresh$;
  - шаг алгоритма $SMSS$;
  - тайм-аут RTO;
- За каждую итерацию планирования на ресурс посылается $min(cwnd, rwnd) - FlightSize$ задач

Алгоритмы Slow Start и Congestion Avoidance

- Slow Start
  - Метапланировщик увеличивает размер окна *cwnd* на *SMSS* задач для каждого успешного подтверждения со стороны узла EndJob, подтверждающего выполнение задачи
- Congestion Avoidance
  - Для обновления значений *cwnd* ведется подсчет новых задач, которые были подтверждены EndJob. Когда число подтвержденных задач достигнет значения *cwnd*, размер окна *cwnd* может быть увеличен на величину *SMSS* задач.

Алгоритмы Slow Start и Congestion Avoidance

- Когда метапланировщик обнаруживает подтвержение после возникновения тайм-аута RTO, для переменной $ssthresh$ устанавливается значение $max(FlightSize/2, 2 * SMSS)$, а размер окна насыщения $cwnd$ устанавливается в значение $SMSS$
- При обнаружении неудачного выполнения задачи, для $ssthresh$ установливается значение $max(FlightSize/2, 2 * SMSS)$, а для $cwnd$ значение $max(FlightSize/2, IW = cwnd_0)$

Тайм-аут RTO (RFC 793)

- После получения EndJob измеряется величина RTT, раное сумме времени, затраченного на транспортировку задачи, времени ожидания задачи и времени выполнения задачи.
- $SRTT = (ALPHA * SRTT) + ((1 - ALPHA) * RTT)$
- $RTO = min[UBOUND, max[LBOUND, (BETA * SRTT)]]$
- UBOUND задает верхний предел значения тайм-аута (например, 1 час), LBOUND — нижний предел (например, 1 минута), ALPHA — весовой фактор (например, 0.9), а BETA — коэффициент вариаций задержки (например, 2.0).

Сравнение со стандартным алгоритмом GridWay

Алгоритм	Время, мин
GridWay (1600 задач по 1 сек)	208
CC (1600 задач по 1 сек)	82
GridWay (160 задач по 15-20 мин)	196
CC (160 задач по 15-20 мин)	199
GridWay (3200 задач по 1 сек, 2 клиента)	> 500
CC (3200 задач по 1 сек, 2 клиента)	124
GridWay (320 задач по 15-20 мин, 2 клиента)	> 500
CC (320 задач по 15-20 мин, 2 клиента	343

Таблица: Сравнение алгоритмов планирования

sscc.grid.pp.ru

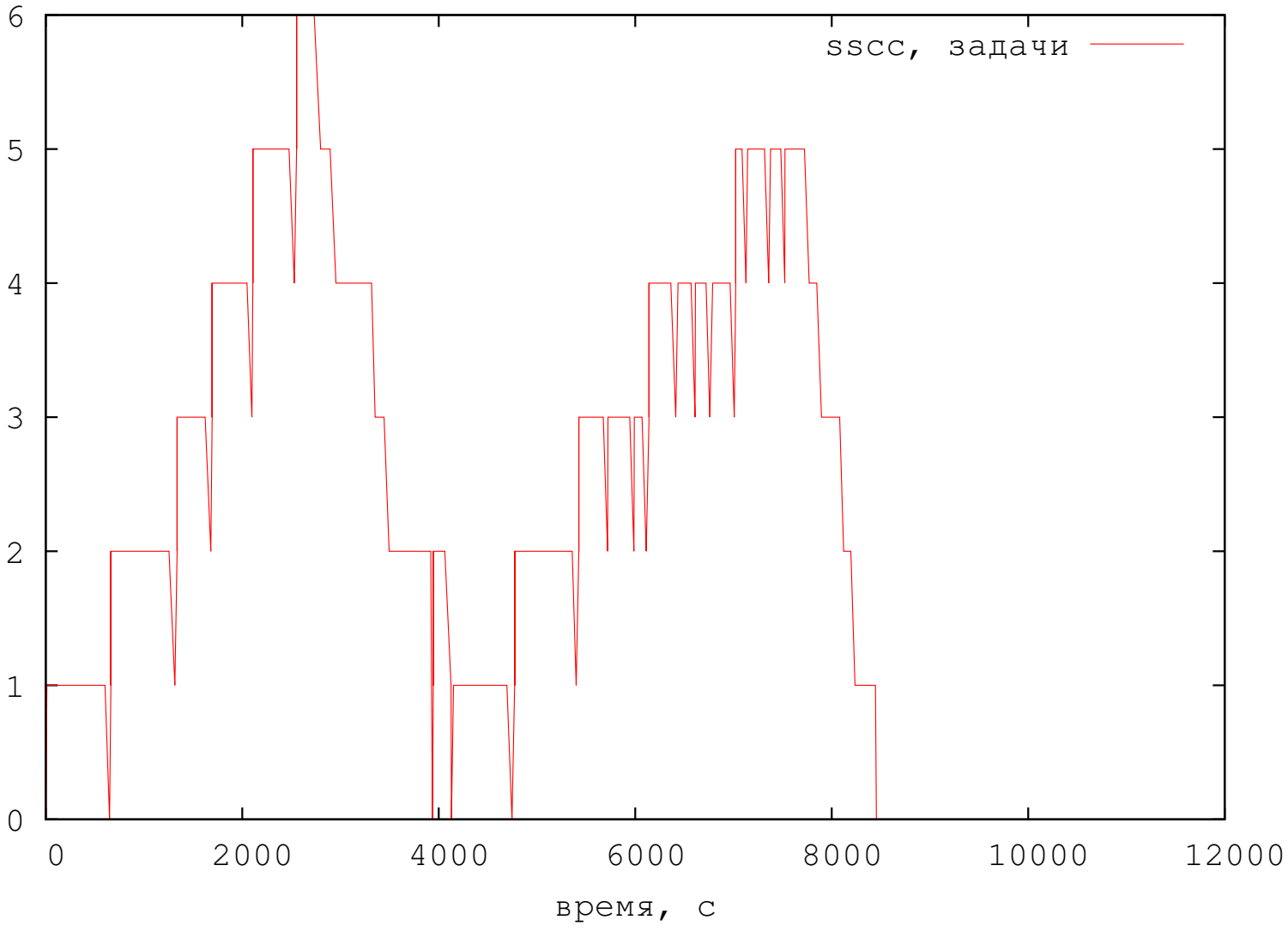


Рис.: Количество запущенных задач на sscc.grid.pp.ru

Системой массового обслуживания GI/G/M(SI,MI,PI,DC,D,C)

1. В любой момент времени общее количество ресурсов системы постоянно, что соответствует $N_i = D = const$, $DI = 0$. (В случае в `sscc` $D = 166$.)

2. За одну итерацию планирования SI (в момент временни iSI) из очереди извлекается $cwnd_i$ задач, где $cwnd_i$ определяется в соответствии с алгоритмом планирования с контролем насыщения.
3. В любой момент времени число обслуживаемых задач не превышает C. (В случае в `sscc` $C = 5$.)

Имитационная модель алгоритма (ALPHA = 0.9 и BETA = 2)

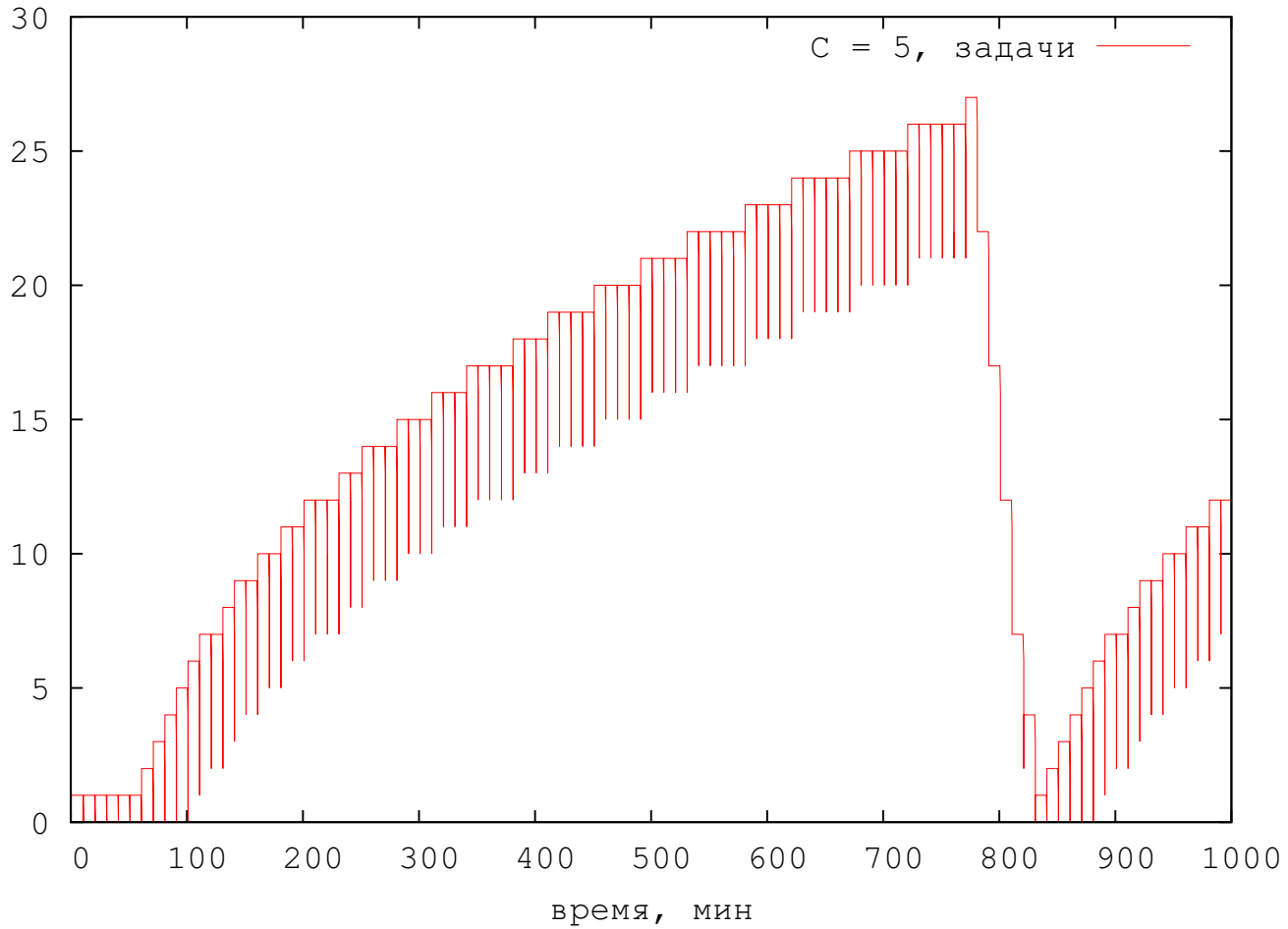


Рис.: Имитационная модель алгоритма планирования с контролем насыщения

Имитационная модель алгоритма (ALPHA = 0.9, BETA = 1.3)

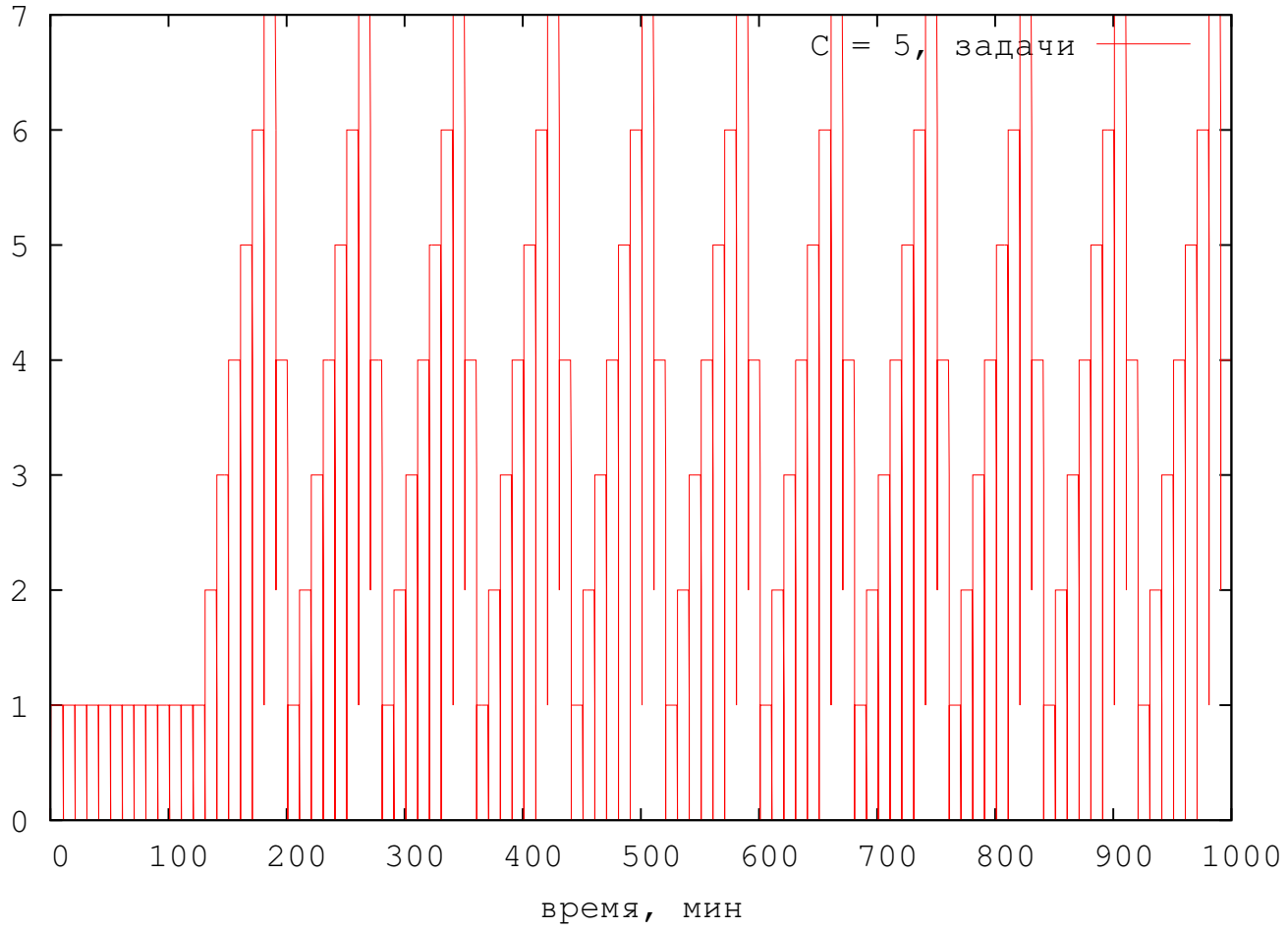


Рис.: Имитационная модель алгоритма планирования с контролем насыщения с параметрами ALPHA = 0.9 и BETA = 1.3

Экспериментальная проверка алгоритма (ALPHA = 0.9, BETA = 1.3)

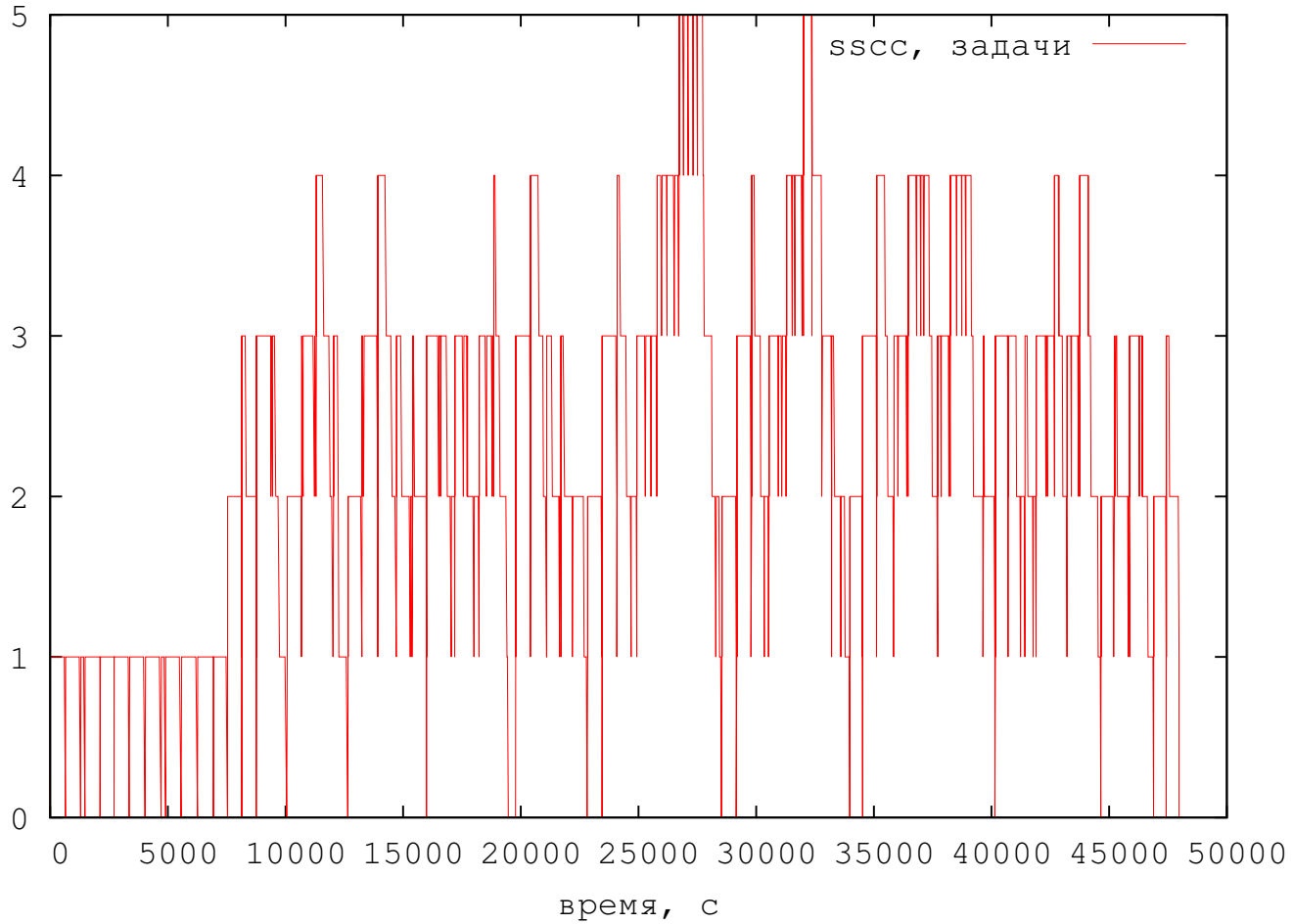


Рис.: Экспериментальная проверка алгоритма планирования с контролем насыщения с параметрами ALPHA = 0.9 и BETA = 1.3

Направления дальнейших исследований

- Оптимизация параметров имитационной модели
- Отказ от *rwnd*
- Децентрализованная диспетчеризация