Представление вычислительного проекта Desktop Grid как системы массового обслуживания

Евгений Евгеньевич Ивашко,
Наталия Николаевна Никитина

Федеральный исследовательский центр
«Карельский научный центр РАН»
Институт прикладных математических исследований
г. Петрозаводск, Россия

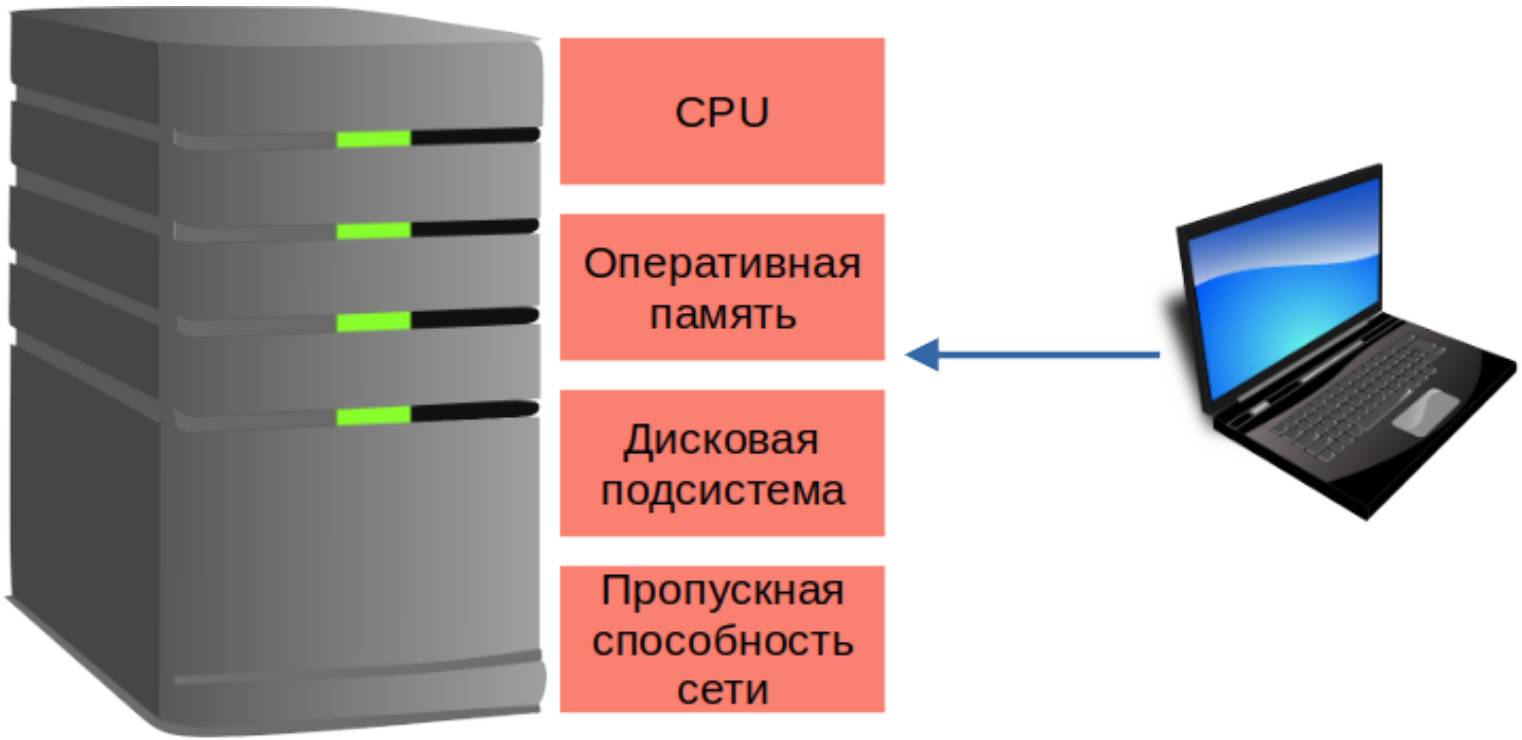
23 сентября 2024 г.

Масштабируемость Desktop Grid

Высокая масштабируемость – существенное преимущество Desktop Grid по сравнению с другими инструментами высокопроизводительных вычислений. Поскольку сервер занимается только взаимодействием с клиентами, производительность Desktop Grid зависит от числа клиентов и их мощности, и в меньшей степени – от мощности самого сервера как устройства, способного обрабатывать запросы.

Масштабируемость Desktop Grid

Ограничением масштабируемости является число запросов в единицу времени, которое может обработать сервер Desktop Grid. Оно зависит от числа клиентов, трудоемкости обработки одного запроса клиента и средней трудоемкости времени расчета одного задания.



Масштабируемость Desktop Grid

Примеры методов снижения нагрузки на сервер Desktop Grid

- Пауза перед повтором запроса к серверу в случае отсутствия ответа, реализованная в BOINC *(дополнительная нагрузка на сервер, простой вычислительных узлов и, как следствие, потеря производительности системы)*.
- Группировка узлов Desktop Grid с наличием промежуточной подсистемы распределения заданий в каждой группе.
- Группировка вычислительных заданий в пакеты.

Какую нагрузку может выдержать сервер Desktop Grid (сколько вычислительных узлов поддерживать)?

В данной работе предлагается модель Desktop Grid как СМО и вычисляются ее характеристики. Метод может использоваться для оценки параметров производительности Desktop Grid.

Математическая модель

Рассмотрим Desktop Grid, в которой k вычислительных узлов и N незавершенных вычислительных заданий, где $N \to \infty$.

Заявка – это обращение клиента к серверу Desktop Grid с отчетом о выполненных заданиях и запросом новых заданий. Веб-сервер использует дисциплину обслуживания FIFO, имеется буфер ожидания (очередь) размером m.

Без потери общности будем полагать, что времена выполнения отдельных заданий и, соответственно, запросов к серверу, являются независимыми одинаково распределенными случайными величинами, распределенными экспоненциально с параметром λ, а времена обслуживания – с параметром μ; $\rho = \lambda/\mu < 1$.

Если в момент прихода заявки сервер свободен или в буфере ожидания есть место, то заявка попадает в систему. Если нет – она отбрасывается, при этом клиент Desktop Grid входит в режим ожидания, чтобы впоследствии повторно обратиться к серверу.

Такая модель представляет Desktop Grid как СМО с ограниченной очередью $M/M/1/m$, описываемую процессом рождения-гибели.

Математическая модель

Система свободна и готова выполнить заявку	$p_0 = \frac{1-\rho}{1-\rho^{m+2}}$
Система занята, а в очереди i заявок, $i = 0, 1, ..., m$	$p_i = \rho^i \cdot p_0 = \rho^i \cdot \frac{1-\rho}{1-\rho^{m+2}}$
Заявка получит отказ	$P_{den} = \rho^{m+1} p_0 = \rho^{m+1} \frac{1-\rho}{1-\rho^{m+2}}$
Заявка будет обслужена	$P_{serv} = 1 - P_{den} = \frac{1-\rho^{m+1}}{1-\rho^{m+2}}$
Среднее число заявок на обслуживании (загрузка канала)	$C = 1 - p_0 = \frac{\rho-\rho^{m+2}}{1-\rho^{m+2}}$
Средняя длина очереди	$L = \rho^2 \cdot \frac{1-\rho^m(m+1-m\cdot\rho)}{(1-\rho^{m+2})(1-\rho)}$
Среднее число заявок в системе	$T = C + L = \frac{\rho-\rho^{m+2}}{1-\rho^{m+2}} + \rho^2 \cdot \frac{1-\rho^m(m+1-m\cdot\rho)}{(1-\rho^{m+2})(1-\rho)}$
Пропускная способность системы (отн.)	$Q = 1 - \rho^{m+1} p_0 = \frac{1-\rho^{m+1}}{1-\rho^{m+2}}$
Пропускная способность системы (абс.)	$A = \lambda(1 - \rho^{m+1} p_0) = \lambda \frac{1-\rho^{m+1}}{1-\rho^{m+2}}$
Среднее время пребывания заявки в очереди	$W_l = \frac{L}{\lambda} = \frac{\rho^2}{\lambda} \cdot \frac{1-\rho^m(m+1-m\cdot\rho)}{(1-\rho^{m+2})(1-\rho)}$
Среднее время пребывания заявки под обслуживанием	$W_p = \frac{C}{\lambda} = \frac{\rho-\rho^{m+2}}{\lambda(1-\rho^{m+2})}$
Среднее время пребывания заявки в системе	$W_s = \frac{T}{\lambda} = \frac{\rho-\rho^{m+2}}{1-\rho^{m+2}} + \rho^2 \cdot \frac{1-\rho^m(m+1-m\cdot\rho)}{\lambda(1-\rho^{m+2})(1-\rho)}$

Характеристики СМО на примере BOINC-проекта SiDock@home

Проект SiDock@home, посвященный виртуальному скринингу лекарств, работает с 2020 г. В ходе BOINC-соревнований число активных компьютеров доходит до 10000, и интенсивность запросов к серверу сильно возрастает. Поэтому актуальна задача оценки нагрузки, выдерживаемой сервером.

Вычислим характеристики СМО на основе статистики проекта SiDock@home за неделю работы (1698 активных компьютеров, приславших хотя бы два обращения к серверу за данную неделю).

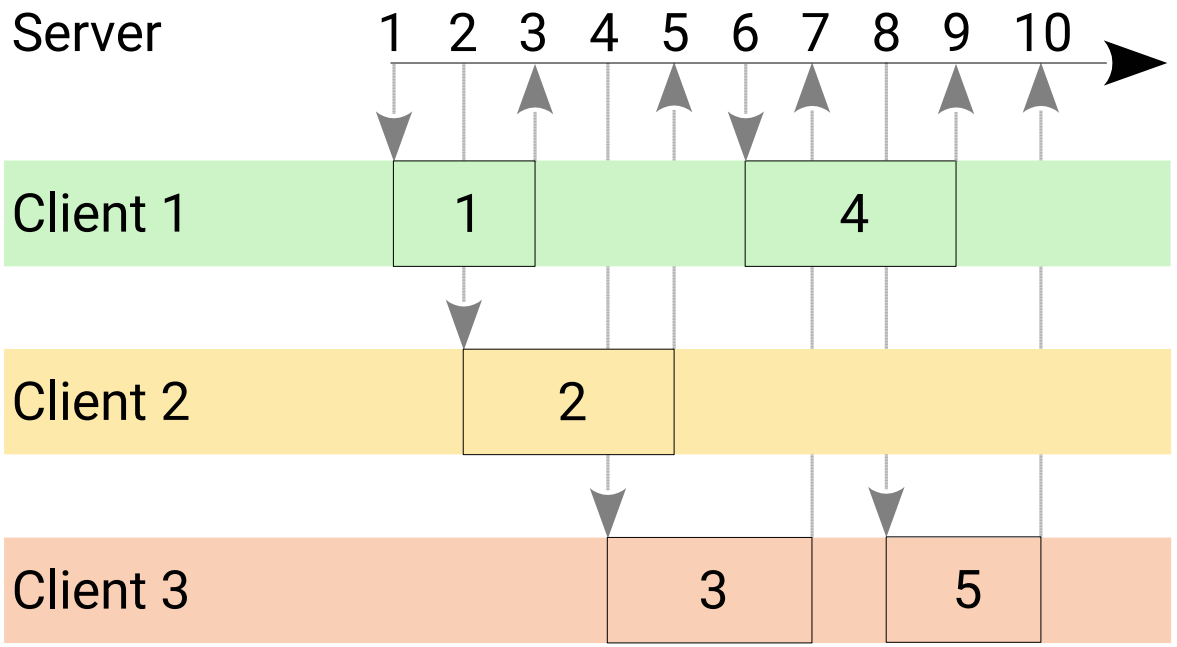
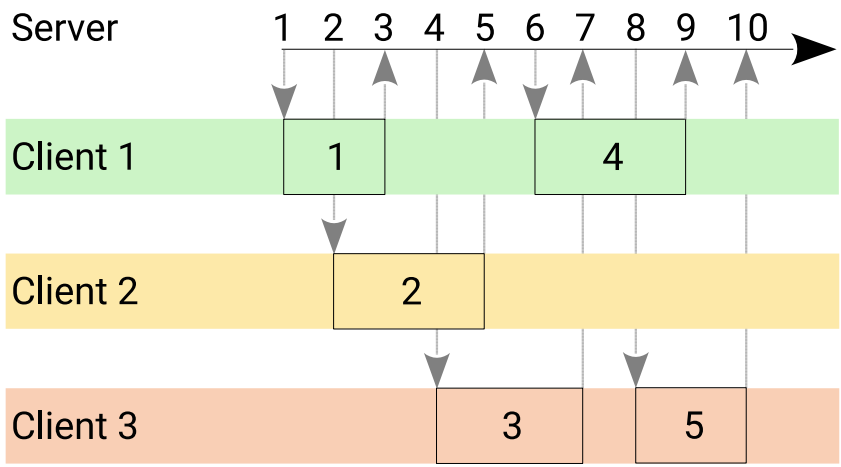


Рис. 1: Пример: последовательность событий при обработке 5 заданий тремя BOINC-клиентами.

t	Клиент	Задание	Тип события BOINC-сервера	Тип HTTP-запроса
1	1	1	sent	GET
2	2	2	sent	GET
3	1	1	received	POST
4	3	3	sent	GET
5	2	2	received	POST
6	1	4	sent	GET
7	3	3	received	POST
8	2	5	sent	GET
9	1	4	received	POST
10	2	5	received	POST



Интенсивность приходов заявок λ

Таблица `result` в базе данных проекта содержит информацию о жизненном цикле каждого экземпляра BOINC-заданий (поля `result.create_time`, `result.sent_time`, `result.received_time` – моменты соответствующих событий).

Приходы заявок происходят в моменты `result.received_time`. С точки зрения сервера, интенсивность приходов заявок составляет $\lambda = 0.070782828$. Интенсивность приходов заявок от «усредненного клиента» составляет $\lambda_{avg} = \lambda/1698 = 0.000041686$.

Используем найденные значения для оценки характеристик СМО и прогноза их изменения в зависимости от изменения числа клиентов.

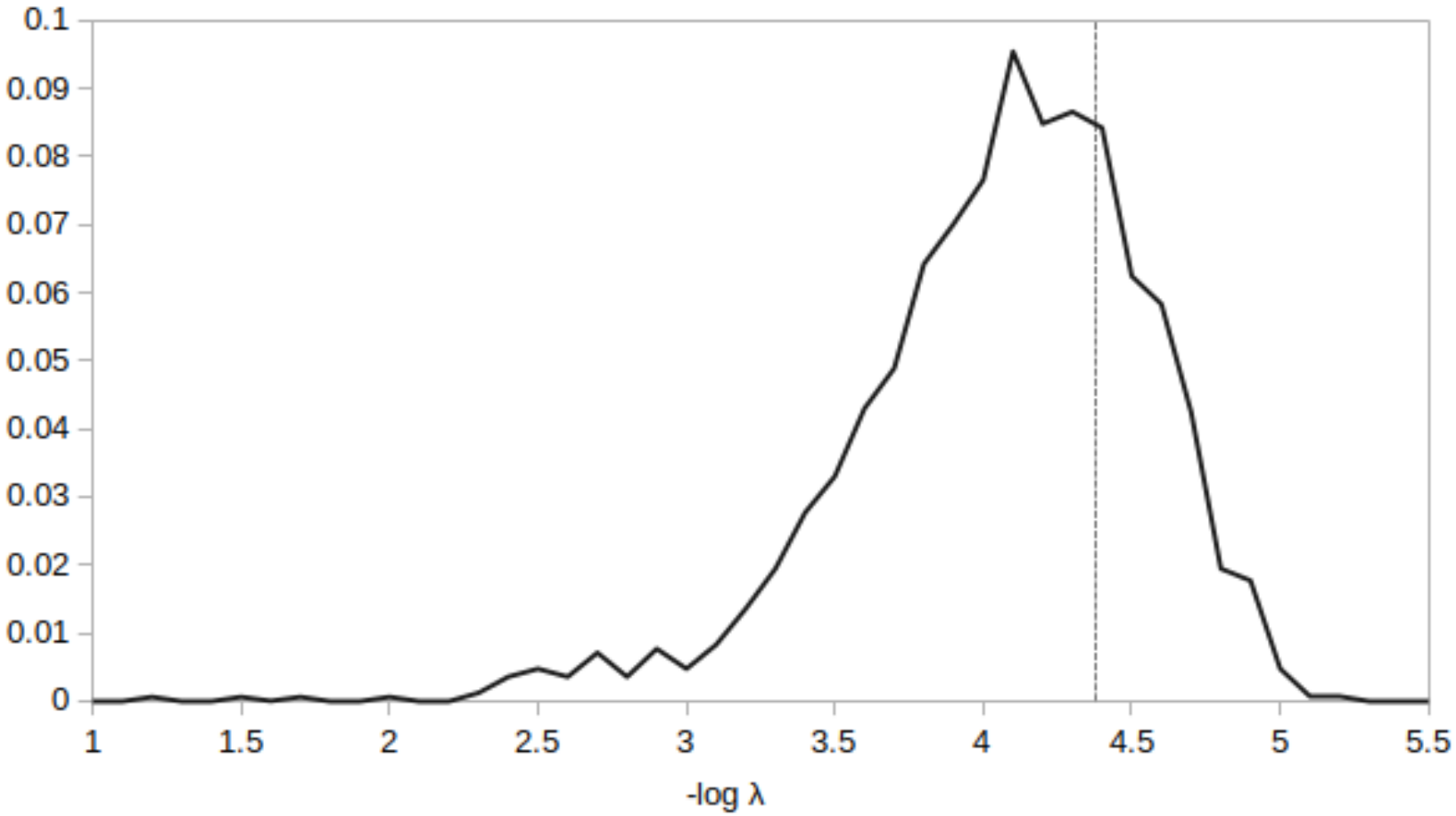


Рис. 2: Эмпирическая плотность распределения интенсивностей приходов заявок от отдельных клиентов. Пунктирная линия соответствует значению λ_{avg} $(-\log \lambda_{avg} = 4.38001)$ для «усредненного клиента».

Интенсивность потока обслуживания заявок μ

Обслуживание заявки происходит на сервере с момента прихода от BOINC-клиента результатов до момента отправки клиенту новых заданий (в лог-файле веб-сервера Apache – соответственно, HTTP-запрос типа POST к процессу `cgi` или `file_upload_handler` и последующий HTTP-запрос типа GET).

Поскольку BOINC-клиент может не запросить новые задания сразу (например, потому что хозяин компьютера запретил их запрашивать, или потому что компьютер занят другими проектами), введем порог τ и отбросим события с длительностью обслуживания больше τ.

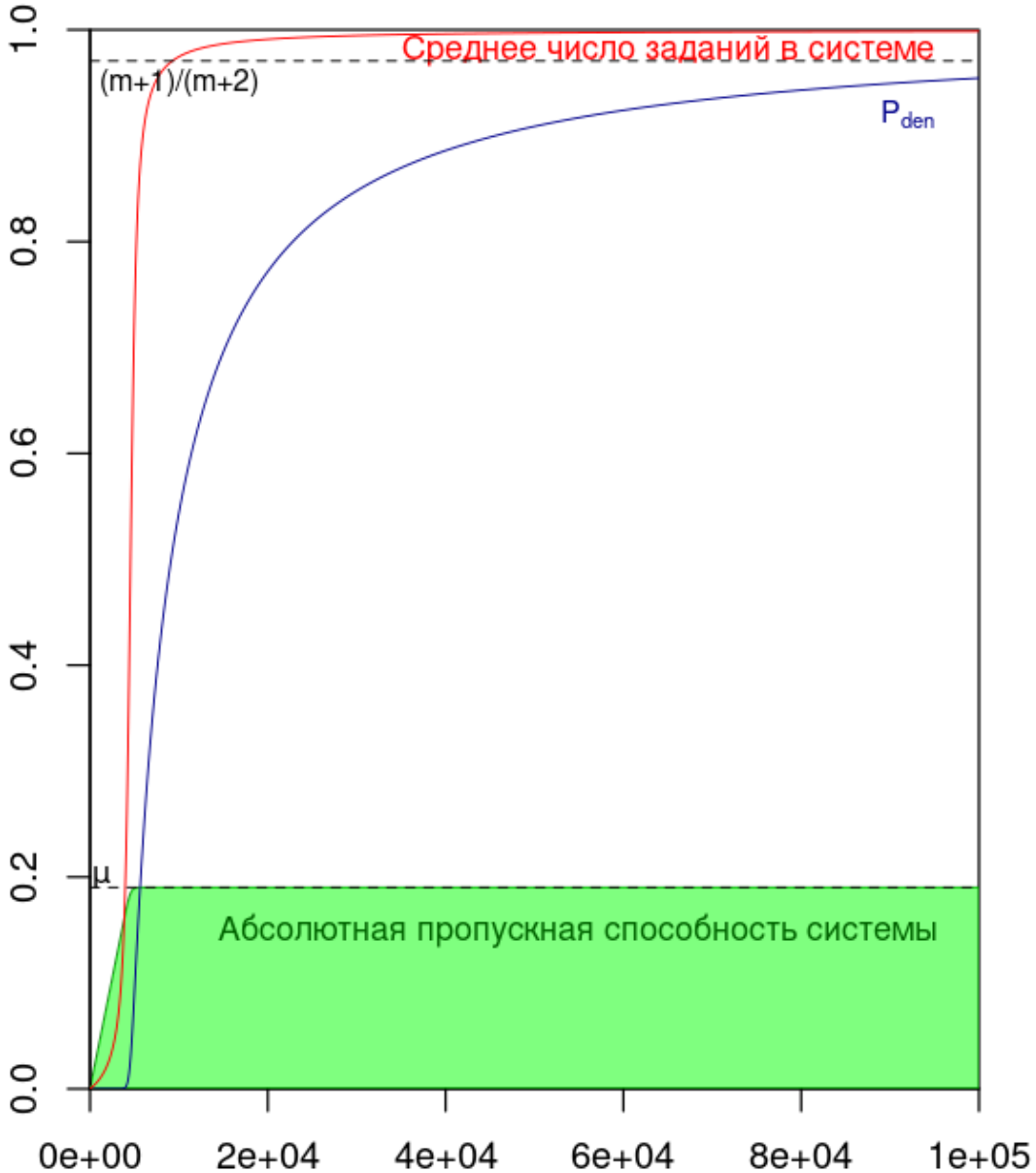
Интенсивность потока обслуживания составляет $\mu = 0.191$ (τ=1 час).

Размер буфера ожидания m

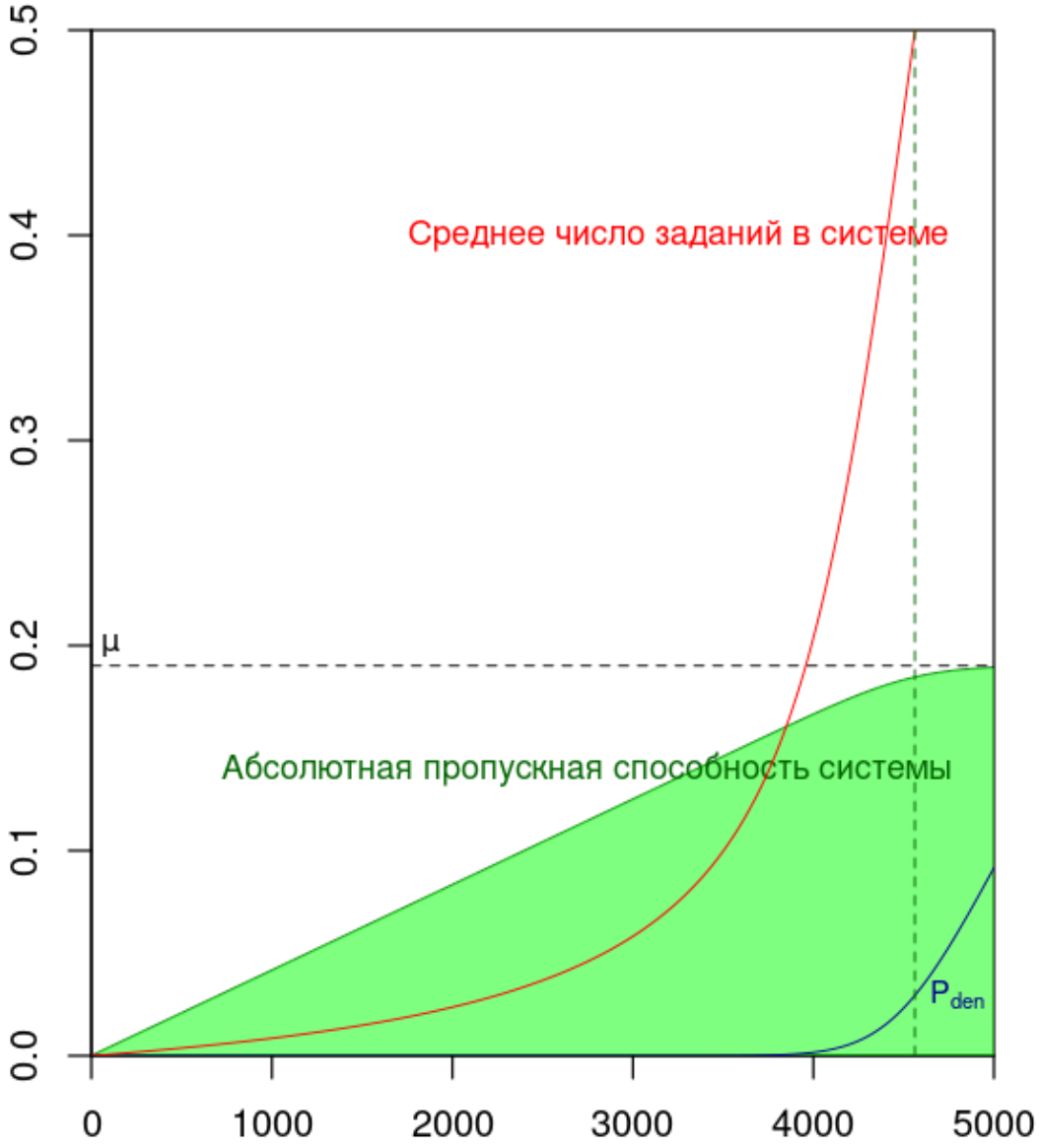
Каждый HTTP-запрос порождает отдельный процесс на веб-сервере Apache. Число активных процессов ограничено директивой конфигурации Apache `ServerLimit 32`, поэтому будем считать, что $m = 32$.

Характеристика	Значение
Число клиентов	$k = 1968$
Интенсивность приходов	$\lambda = 0.071$
Интенсивность обслуживания	$\mu = 0.190$
Емкость буфера ожидания	$m = 32$
Коэффициент загрузки системы	$\rho = \lambda/\mu = 0.372$
Вероятность пустой системы	$p_0 = 0.628$
Вероятность отказа	$P_{den} = 4.298 \times 10^{-15}$
Вероятность получить обслуживание	$P_{serv} = 0.999$
Среднее число заявок на обслуживании	$C = 0.372$
Среднее число заявок в очереди	$L = 0.221$
Среднее число заявок в системе	$T = 0.593$
Пропускная способность системы (отн.)	$Q = 0.999$
Пропускная способность системы (абс.)	$A = 0.071$
Среднее время пребывания заявки в очереди	$W_l = 3.117$
Среднее время обслуживания заявки	$W_p = 5.258$
Среднее время пребывания заявки в системе	$W_s = 8.376$

Таблица 1: Характеристики системы *M/M/1/m* для SiDock@home.



Производительность системы при числе клиентов $k \leq 100000$.



Производительность системы при числе клиентов $k \leq 5000$.

Характеристики системы $M/M/1/m$ для SiDock@home

- Низкий коэффициент загрузки системы $\rho \approx 0.37$ и относительно высокая вероятность того, что система свободна и готова выполнить заявку $p_0 \approx 0.63$.
- Вероятность отказа близка к нулю ($P_{den} \approx 4 \cdot 10^{-15}$), и вероятность получить обслуживание близка к единице ($P_{serv} = 0.(9)$).
- Абсолютная пропускная способность системы равна интенсивности входного потока заявок ($A \approx 0.07$).

При текущих настройках и параметрах сервера проект может поддерживать вдвое больше вычислительных узлов. При $\sim$4500 активных узлах система достигнет максимальной абсолютной пропускной способности с низкой вероятностью отказа $P_{den} \approx 0.03$.

Повышение масштабируемости и производительности Desktop Grid

- Добавить один или несколько серверов $M/M/1/m \rightarrow M/M/k/m$.
- Изменить настройки веб-сервера и/или BOINC-сервера.
- Изменить интенсивность приходов заявок, меняя продолжительность выполнения заданий (например, группируя маленькие задания).

Ограничения предложенного подхода

- Предложенная математическая модель основана на гипотезах об экспоненциальном распределении времени между приходами заявок и времени обслуживания. Для применения на практике необходимо проверить гипотезы на достаточно больших выборках данных. Если распределения другие, то характеристики системы будут выражаться по-другому.
- Распределения могут меняться при изменении характера нагрузки на сервер (например, в периоды соревнований).
- Для подготовки выборки данных требуется специальная подготовка сервера проекта (формат лог-файлов и отладочных сообщений, работа с базой данных).

Преимущества предложенного подхода

- Аналитические выражения характеристик Desktop Grid позволяют оценить масштабируемость и производительность системы, могут быть использованы в случае экспоненциального распределения и модифицированы для ряда других распределений.
- Математическая модель позволяет спрогнозировать различные изменения в производительности системы, связанные с изменениями входящего потока заявок, настроек сервера, параметров вычислительной задачи и т.д.
- Математическая модель Desktop Grid как СМО позволяет сравнивать Desktop Grid с другими СМО и применять методы теории массового обслуживания.
- Возможно автоматизировать расчет характеристик Desktop Grid и формирование рекомендаций по оптимизации.

Спасибо за внимание!

E-mail: nikitina@krc.karelia.ru