Комплексный подход к анализу истории выполнения и выбору параметров запуска суперкомпьютерных приложений на основе данных системного мониторинга

Паокин Андрей Викторович

andrejpaokin@yandex.ru

Никитенко Дмитрий Александрович

dan@parallel.ru

Научно-исследовательский вычислительный центр МГУ имени М.В.Ломоносова

“Суперкомпьютерные дни в России”, г. Москва, 23 сентября 2024

Актуальность анализа эффективности

Суперкомпьютеры — **дорогой, востребованный** и **сложный** вычислительных ресурс

- Конкуренция за ресурсы может привести к повышенному времени ожидания задачи в очереди на счёт
- Время до получения результата = время ожидания в очереди на счёт + время выполнения. **Выбор параметров** задачи (раздел, запрошенные времена счёта и число узлов, ...) влияет на времена ожидания в очереди на счёт и выполнения как для отдельной задачи, так и для всех задач суперкомпьютера
- Контроль доступа, отчётность, техническая поддержка, ...

Необходим анализ как на уровне **всего потока задач**, так и на уровне **отдельных пользователей**

Существующие инструменты и источники данных. TASC

TASC (Tuning Applications for SuperComputers) — инструмент, позволяющий выявлять проблемы с эффективностью суперкомпьютерных приложений.

Используемые данные:

- Данные о задаче из **планировщика**:
  - раздел, аккаунт, статус, запрошенные число узлов и время счёта, времена постановки задачи в очередь, работы и окончания
- Характеристики **работы приложений** с поминутной агрегацией
  - загрузка процессора и CPU, loadavg, счетчик инструкций, кэш-промахи, объём переданных данных по сети, ...

Пример проблем с эффективностью:

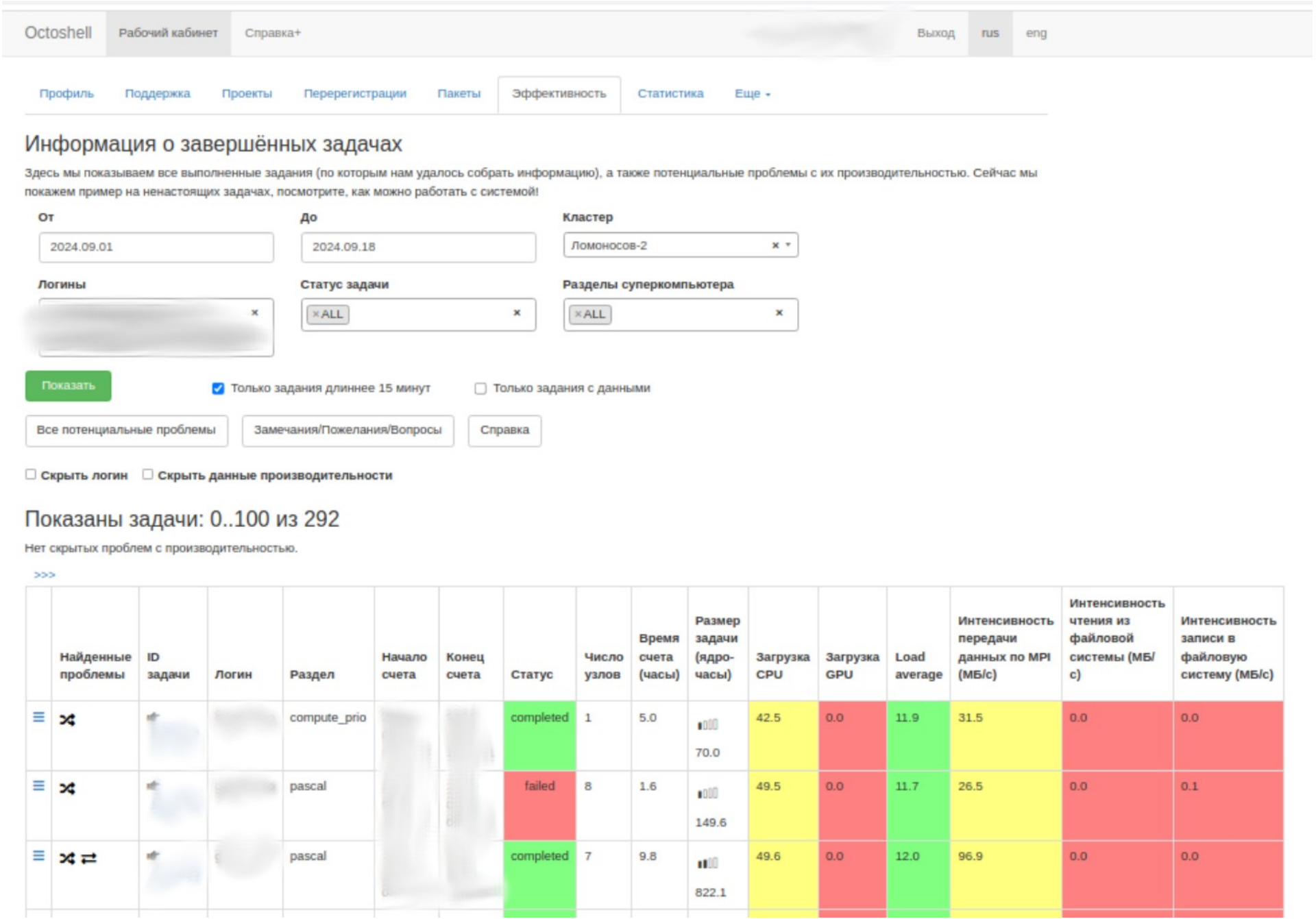
- Одноядерная задача
- Низкая активность использования всех доступных ресурсов
- Низкая сетевая локальность при активном использовании сети

Существующие инструменты и источники данных. Octoshell

Octoshell — система поддержки функционирования суперкомпьютерных центров

- Модульное веб-приложение на отдельном от суперкомпьютера физическом сервере с открытым исходным кодом
- Единный интерфейс для организации рабочих процессов:
  - контроль доступа на вычислительные системы, профиль проекта, техническая поддержка, перерегистрации (отчётность), эффективность приложений, рассылки, виртуальные машины и др.
- Для организации, закрытия и возобновления доступа на вычислительные системы Octoshell выполняет команды по SSH
- Используется в СКЦ МГУ и ведутся работы по внедрению в других суперкомпьютерных центрах (создание подробной документации, конфигурирование при установке)

Анализ эффективности в Octoshell



Octoshell | Рабочий кабинет | Справка+ | Выход | rus | eng

Профиль | Поддержка | Проекты | Перерегистрации | Пакеты | Эффективность | Статистика | Еще

Информация о завершённых задачах

Здесь мы показываем все выполненные задания (по которым нам удалось собрать информацию), а также потенциальные проблемы с их производительностью. Сейчас мы покажем пример на ненастоящих задачах, посмотрите, как можно работать с системой!

От: 2024.09.01 | До: 2024.09.18 | Кластер: Ломоносов-2

Логины | Статус задачи: ALL | Разделы суперкомпьютера: ALL

Показать | Только задания длиннее 15 минут | Только задания с данными

Все потенциальные проблемы | Замечания/Пожелания/Вопросы | Справка

Скрыть логин | Скрыть данные производительности

Показаны задачи: 0..100 из 292

Нет скрытых проблем с производительностью.

>>>

	Найденные проблемы	ID задачи	Логин	Раздел	Начало счета	Конец счета	Статус	Число узлов	Время счета (часы)	Размер задачи (ядро-часы)	Загрузка CPU	Загрузка GPU	Load average	Интенсивность передачи данных по MPI (МБ/с)	Интенсивность чтения из файловой системы (МБ/с)	Интенсивность записи в файловую систему (МБ/с)
≡	⤮			compute_prio			completed	1	5.0	70.0	42.5	0.0	11.9	31.5	0.0	0.0
≡	⤮			pascal			failed	8	1.6	149.6	49.5	0.0	11.7	26.5	0.0	0.1
≡	⤮ ⇄			pascal			completed	7	9.8	822.1	49.6	0.0	12.0	96.9	0.0	0.0

Свойства разрабатываемого подхода к анализу история выполнения

- Ориентирован на различные роли пользователей (руководитель проекта, простой участник проекта, перерегистратор, техподдержка и др.)
- Главный экран с кратким описанием текущей ситуации
  - Распределение последних задач по статусам завершения, интенсивность запусков
  - Времена выполнения и ожидания, перезаказ ресурсов (где время выполнения << запрошенное время)
  - Статистика использования ресурсов и пакетов
  - Место проекта/отдельного пользователя по эффективности, интенсивности запусков
- Постепенный переход к интересующим подмножествам задач
  - Например, Задачи со статусом TIMEOUT в разделе compute → задачи с плохой сетевой локальностью → задачи, задействовавшие более 30 узлов

Свойства разрабатываемого подхода к анализу история выполнения

- Выделение подмножеств запусков и выбор свойств для сравнения осуществляется совместно пользователем и системой, реализующей подход
- Методы анализа:
  - Анализ серий (хронологий) запусков
    - В какой момент появились проблемы с эффективностью?
  - Сравнение отдельного запуска и группы похожих запусков
    - Чем запуск отличается от группы похожих запусков?
  - Сравнение групп запусков
    - В чем одинаковы и отличаются группы запусков?
- Создание правил для снятия неэффективных задач

Время ожидания задачи в очереди

Время до получения результата = время ожидания в очереди на счёт + время выполнения. Задача предсказания времени выполнения приложения на данный момент не имеет решения. А что если **можно оценивать время ожидания** задачи в очереди?

Существующие подходы:

- Встроенные в планировщики (например, в SLURM)
  - Возможность получить оценку без постановки задачи в очередь
  - Предполагается, что все задачи будут выполняться столько, сколько пользователь указал в лимите
- Подходы, основанные на машинном обучении
  - Планировщик как «чёрный ящик»
  - Задача регресии или классификации
  - Признаковое представление: данные о задаче и очереди
    - Представление очереди (вектора нефиксированной длины) → агрегирующие функции и создание гистограмм после сегментирования (data binning)

Представление очереди. Пример

Пусть в определённый момент очереди находится 7 ожидающих запуска задач, которые запросили следующие количество узлов:

[2, 4, 6, 7, 11, 17, 34]

В среднем в очереди 11.57 узлов, **минимум** — 2, **максимум** — 34.

Дополним гистограммой из 4 элементов, применяя разные методы для формирования гистограмм:

- Биннинг с равными отрезками → смещение, слишком низкий разброс на некоторых подинтервалах

(1, 9]	(10,18]	(18,26]	(26, 34]
4	2	0	1

- Равночастотный биннинг → разбиение может плохо отражать свойства предметной области

(1,5]	(5,7]	(7, 14]	(14,34]
2	2	1	2

Предложенная модель оценки времени ожидания

В итоге был модифицирован и апробирован подход к описанию объекта, описанный в статье:

N. Brown, G. Gibb, E. Belikov, and R. Nash, “Predicting Batch Queue Job Wait Times for Informed Scheduling of Urgent HPC Workloads,”https://arxiv.org/abs/2204.13543 (2022).

- Авторы оценивают простым способом время выполнения задачи. Пусть оно известно → наилучший возможный результат.
- Предсказываем интервал, в котором находится ожидаемое время выполнения.
- Рассматриваются задачи очередей compute и pascal «Ломоносов-2» с 2019 года
- Алгоритм планирования: FIFO с включённым conservative backfilling, у каждого пользователя есть лимит на число одновременно запущенных задач
- Подбираемые гиперпараметры:
  - число интервалов (бинов)
  - метод биннинга для формирования гистограмм: равночастотный и биннинг с равными отрезками

Представление одного объекта

Признаки задачи и пользователя:

- Число узлов запрошенной задачи
- Временной лимит
- Число задач данного пользователя в очереди
- Лимит пользователя на число одновременно запущенных задач

Состояние запущенных задач:

- Число работающих задач
- Суммарное число узлов, занятых работающими задач
- Суммарная оставшаяся работа запущенных задач
- Гистограмма узлов работающих задач
- Гистограмма оставшейся работы работающих задач
- Гистограмма оставшегося времени работы работающих задач

Состояние очереди ожидающих задач:

- Число задач, ожидающих в очереди
- Суммарное число задач узлов, запрошенных ожидающими в очереди задачами
- Суммарная работа (число узлов на время выполнения) ожидающих в очереди задач
- Медиана времени ожидания задач, ожидающих постановки на счёт
- Гистограмма узлов, запрошенных ожидающими в очереди задачами
- Гистограмма работ ожидающих в очереди задач
- Гистограмма времён ожидания ожидающих задач

Полученные результаты на модели случайного леса в «Pascal»

Parameters		Test sample		Class accuracy						
Bins	Binning method	Accuracy	standard deviation of error	[0 minutes, 1 minute)	[1 minute, 30 minutes)	[30 minutes, 1 hour)	[1 hour, 4 hours)	[4 hours, 8 hours)	[8 hours, 16 hours)	[16 hours, $+\infty$)
-	-	0.7399	1.0573	0.9492	0.1059	0.0237	0.4241	0.3875	0.5331	0.1888
3	freq	0.7545	1.0568	0.9563	0.1314	0.0	0.4739	0.4354	0.5654	0.1674
6	freq	0.7522	1.0638	0.9602	0.1032	0.0	0.4334	0.4705	0.5499	0.1631
6	width	0.7424	1.0872	0.9563	0.0818	0.0079	0.3847	0.4207	0.5752	0.1288
8	freq	0.7524	1.0794	0.9632	0.1166	0.004	0.4357	0.4539	0.5401	0.0987
8	width	0.7443	1.0842	0.9612	0.071	0.0079	0.4009	0.3782	0.5809	0.1245
16	freq	0.7499	1.1051	0.9683	0.0938	0.0	0.3986	0.4004	0.5738	0.0773
16	width	0.7472	1.121	0.9633	0.063	0.0	0.3998	0.476	0.564	0.0515
8	freq (no limit features)	0.7485	1.1295	0.9635	0.0429	0.0079	0.4044	0.4779	0.5767	0.0987

- Представление состояния очереди с помощью биннинга улучшает точность на ~1,5%

Полученные результаты на модели случайного леса в «Compute»

Parameters		Test sample		Class accuracy						
Bins	Binning method	Accuracy	standard deviation of error	[0 minutes, 1 minute)	[1 minute, 30 minutes)	[30 minutes, 1 hour)	[1 hour, 4 hours)	[4 hours, 8 hours)	[8 hours, 16 hours)	[16 hours, $+\infty$)
-	- (since 2019)	0.5902	1.4813	0.7264	0.1216	0.0019	0.2994	0.0448	0.0911	0.9407
8	freq (since 2019)	0.6072	1.5287	0.8282	0.0299	0.0	0.16	0.013	0.0296	0.9748
-	- (2021)	0.614	1.4476	0.6688	0.5388	0.0	0.2659	0.1111	0.1725	0.8719
8	freq (2021)	0.6145	1.576	0.6403	0.5999	0.0	0.1991	0.0145	0.0291	0.9036

Анализ результатов оценки времени ожидания

- Точность предсказаний в Pascal ниже 75% в облегчённых условия → недостаточно высокая, однако примерно такая же, как в других работах
- В очереди Compute точность оказалось значительно меньше, чем на Pascal, возможно из-за того, что:
  - На Compute разброс времени ожидания больше
  - Не учтена очередь compute_prio
  - Изменчивость времени ожидания → дисбаланс между классами в обучающей и контрольной выборке

Сценарий для оценки перезаказа ресурсов

Для одной отдельно взятой задачи можно оценивать перезаказ ресурсов (избыточный запрос ресурсов) простым способом:

Перезаказ ресурсов = запрошенный временной лимит — время выполнения

Перезаказ ресурсов, равный одному часу — это большой перезаказ или нет? Вместо этого перезаказ ресурсов у одной задачи лучше оценить следующим образом:

Перезаказ ресурсов = запрошенный временной лимит **/** время выполнения

Всё множество задач пользователя **разделим на группы** по **числу узлов** и **временному лимиту.** Это можно сделать несколькими способами:

- Группировка по **числу узлов** и **временному лимиту** не сработает для пользователей с большим числом разных параметров запуска
- Разбиение с равными отрезками применим для **временного лимита**

$$a = x_0 < x_1 < ... < x_{n-1} < x_n = b, x_i = a + (b - a)/n * i, i = \overline{0, n}$$

- Разбиение с равными отрезками после логарифмирования для **числа узлов**

$$a = x_0 < x_1 < ... < x_{n-1} < x_n = b, x_0 = a, x_i = a + (b - a)^{i/n}, i = \overline{1, n}$$

Сценарий для оценки перезаказа ресурсов

Разбиение задач пользователя по размеру задачи (число узлов x запрошенный лимит)

Число узлов не меньше

Число узлов не больше

параметр анализа: Превышение запрошенного лимита

Временной лимит не меньше

Временной лимит не больше

Поставлена в очередь не раньше: 2022.01.01 00:00:00

Поставлена в очередь не позднее: 2023.01.01 00:00:00

Команда запуска: Выберите значе…

раздел: compute

Найти

Запрошенное время	Число узлов	Число задач	Доля задач, где запрошенный лимит превышен в 2 раза	Доля задач, где запрошенный лимит превышен в 4 раза
[1, 42)	[1, 3)	25	0.28	0.24
	[3, 13)	105	0.36	0.24
	[13, 49)	25	0.4	0.24
[42, 83)	[1, 3)	26	0.15	0.15
	[3, 13)	60	0.4	0.32

Статистика проекта для перерегистрации

Краткая история использования ресурсов проектом за прошедший год

Сравнение производится с **121** проектами-участниками перерегистрации, которые запустились хотя бы раз на "Ломоносов-2" за прошедший год. Для некоторых пакетов ПО заканчивать свою задачу не в COMPLETED - нормальная ситуация. Места отсортированы от большего к меньшему.

Детали

Состояние	Параметр	Значение	Место по абс.значению	Доля от всех состояний	Место доли
Все	К-во задач	528	№ 47		
	Узлочасы	81675	№ 21		
CANCELLED	К-во задач	27	№ 53	5.11%	№ 46
	Узлочасы	10418	№ 14	12.76%	№ 20
COMPLETED	К-во задач	291	№ 42	55.11%	№ 51
	Узлочасы	3195	№ 57	3.91%	№ 111
FAILED	К-во задач	153	№ 32	28.98%	№ 35
	Узлочасы	35758	№ 9	43.78%	№ 11
NODE_FAIL	К-во задач	1	№ 43	0.19%	№ 29
	Узлочасы	541	№ 12	0.66%	№ 9
TIMEOUT	К-во задач	56	№ 50	10.61%	№ 67
	Узлочасы	31762	№ 25	38.89%	№ 56

Резюме

- Необходимость анализа истории выполнения задач обусловлена дороговизной ресурсов суперкомпьютерных систем, конкуренцией и взаимным влиянием пользователей на времена ожидания задач друг друга
- В СКЦ МГУ разработаны и внедрены в систему Octoshell инструменты, упрощающие анализ выполнения отдельных задач пользователя
- На данный момент разрабатывается подход, позволяющий пользователю анализировать группы запусков и сравнивать их между собой
  - Главный экран, показывающий общую картину с возмжностью спуска к деталям
  - Анализ серий (хронологий) запусков
  - Сравнение отдельного запуска и серий похожих запусков
  - Сравнение серий запусков