ФКН ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

Программный проект на тему:

Разработка подсистемы динамического изменения приоритета вычислительных задач в очереди суперкомпьютера для расширения системы HPC TaskMaster

Выполнил:
Студент 3 курса БПМИИ225 Рыбаков Георгий

Руководитель проекта:
Костенецкий Павел Сергеевич
начальник отдела суперкомпьютерного моделирования
к.ф.-м.н., доцент

Яндекс

HPC и суперкомпьютер «cHARISMa»

Пиковая производительность FP64

2.2 Петафлопс

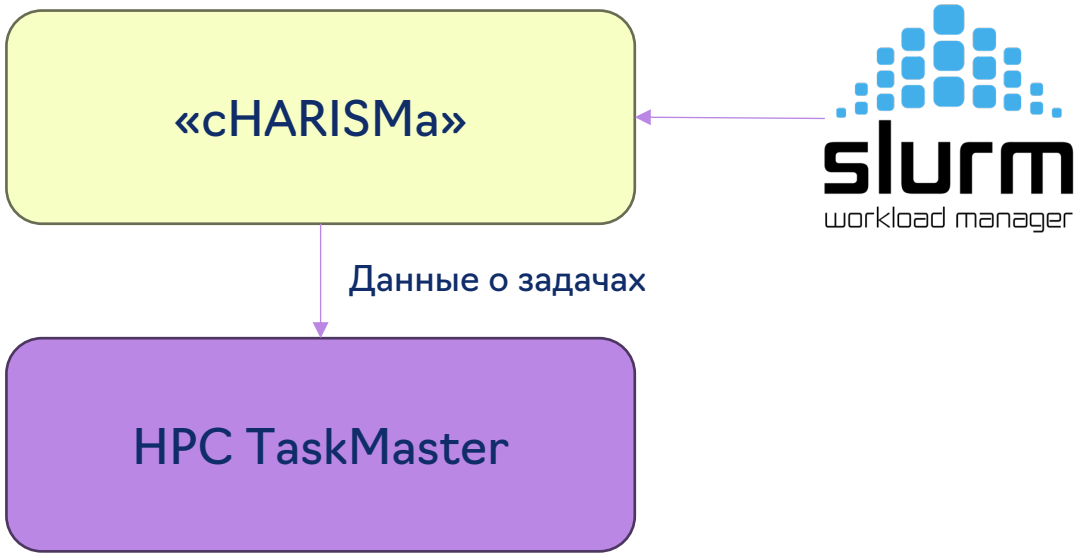
AI производительность

51.1 Петафлопс

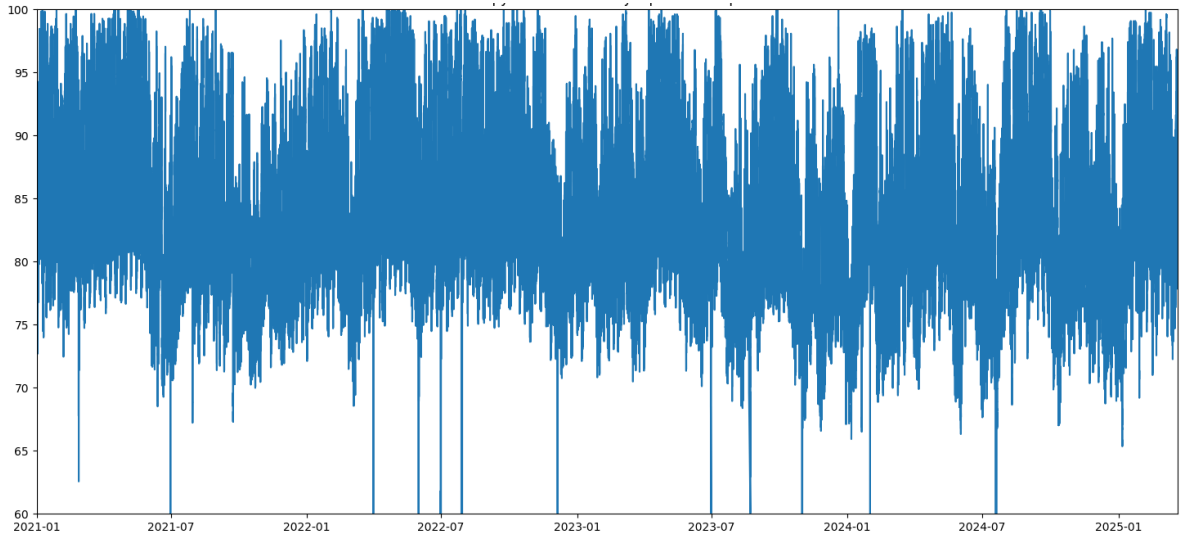
Количество пользователей

>1500

Квоты и приоритеты задач



Загруженность GPU суперкомпьютера



Для эффективного распределения ресурсов «cHARISMa» используется **SLURM**

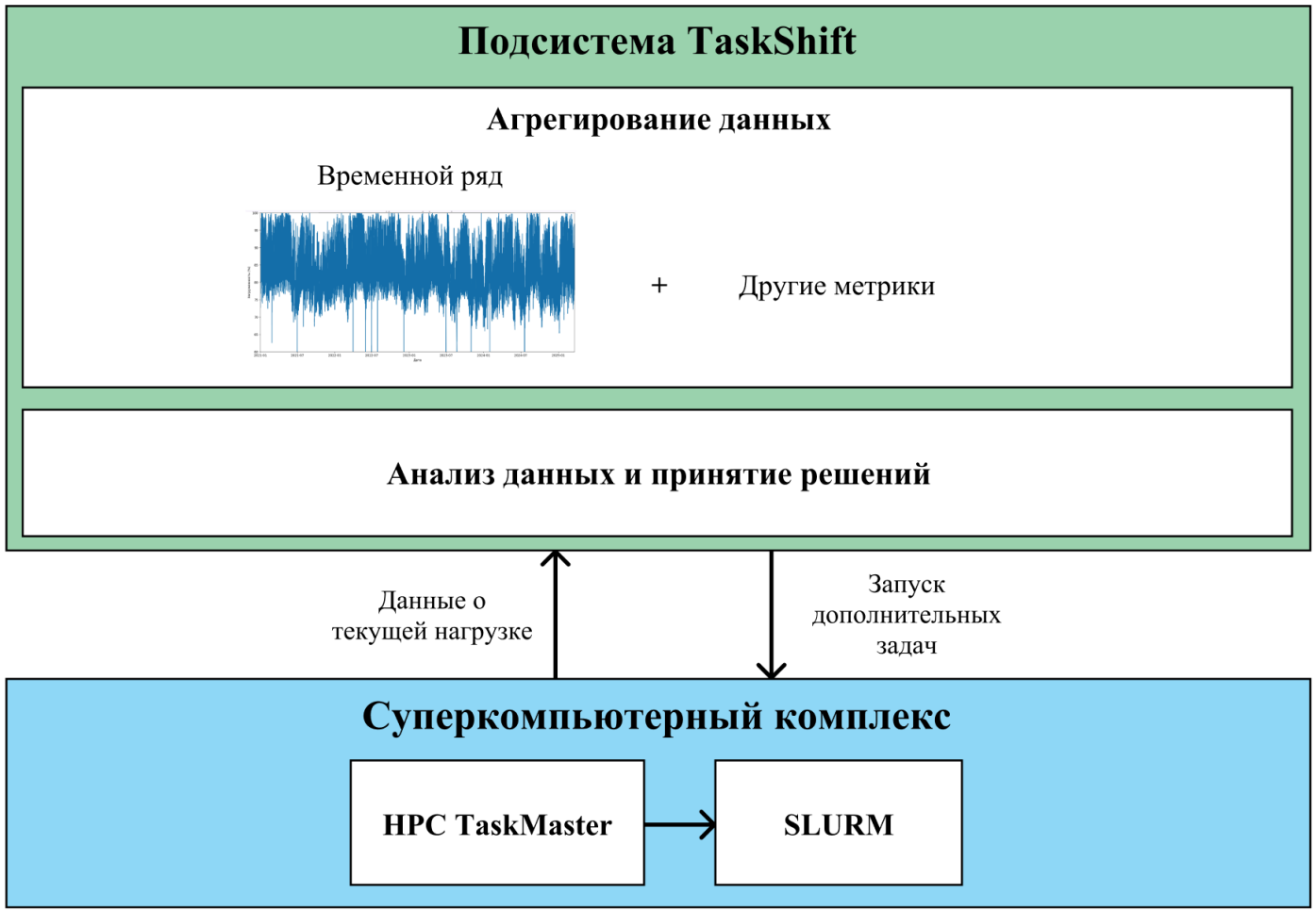
Для мониторинга эффективности - **HPC TaskMaster**.

Приоритет - устанавливаемый алгоритмами SLURM параметр задачи, на основе которого определяется, какая задача будет выполнена первой.

Квоты – ограничения по использованию ресурсов суперкомпьютера, определяемые для каждого пользователя.

Актуальность

Загруженность может изменяться в зависимости от дня недели или времени суток.



Цель и задачи работы

Цель:

Разработать подсистему динамического изменения приоритета вычислительных задач в очереди суперкомпьютера.

Задачи:

- Провести анализ данных загруженности суперкомпьютера.
- Спроектировать архитектуру сервиса.
- Разработать систему повышающую загрузку суперкомпьютера в выходные дни.
- Обеспечить покрытие тестами и простое развёртывание с помощью Docker.

Существующие решения

Так как на суперкомпьютере НИУ ВШЭ используется планировщик задач SLURM, рассматривались доступные решения только для этого планировщика.

Модуль мультифакторной приоритизации учитывает сразу несколько параметров:

$$Job_{priority} = site_{factor} + WeightAge * age_{factor} + WeightFairshare * fairshare_{factor} + WeightJobSize * jobsize_{factor} + ... - nice_{factor}$$

Примеры параметров:

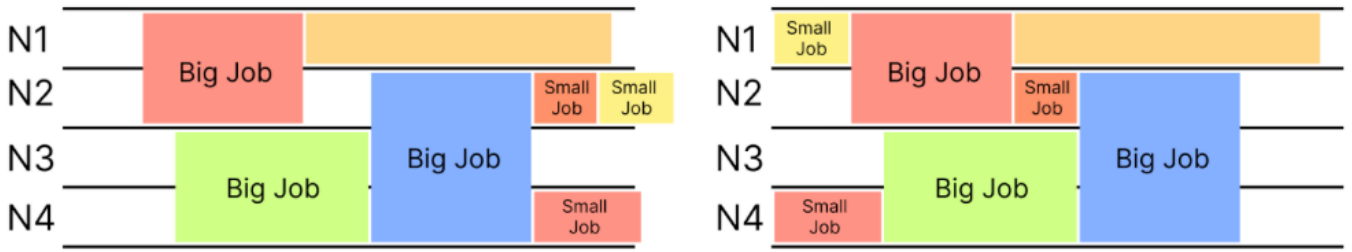
- Age factor - сколько задача находится в очереди.
- QoS - заранее заданные лимиты для каждого пользователя или групп пользователей.
- Fair-Share - признак "честного" распределения, зависящий от того, сколько ресурсов требуется для выполнения задачи и сколько ресурсов уже было занято пользователем.

Минусы:

- Не учитывает внешние факторы, такие как выходные, для определения приоритетов.
- Даже с увеличенным приоритетом, задача может не влезать в доступные квоты пользователя.

Существующие решения

Backfill – позволяет значительно ускорить производительность кластера за счет выполнения маленьких низкоприоритетных задач в промежутке между крупными, если их выполнение не увеличит общее время работы



Минусы:

- Не умеет запускать задачи, которые выходят за квоты пользователей.
- Не учитывает возможную будущую загруженность суперкомпьютера.

Функциональные требования

Подсистема должна:

- собирать данные о загрузке CPU и GPU суперкомпьютера,
- прогнозировать нагрузку с использованием эмпирических методов,
- анализировать очередь задач и выявлять задачи, которые можно запустить в предсказанные интервалы свободной загрузки,
- работать только в определенные моменты времени,
- функционировать как отдельный сервис на главном узле кластера «cHARISMa».

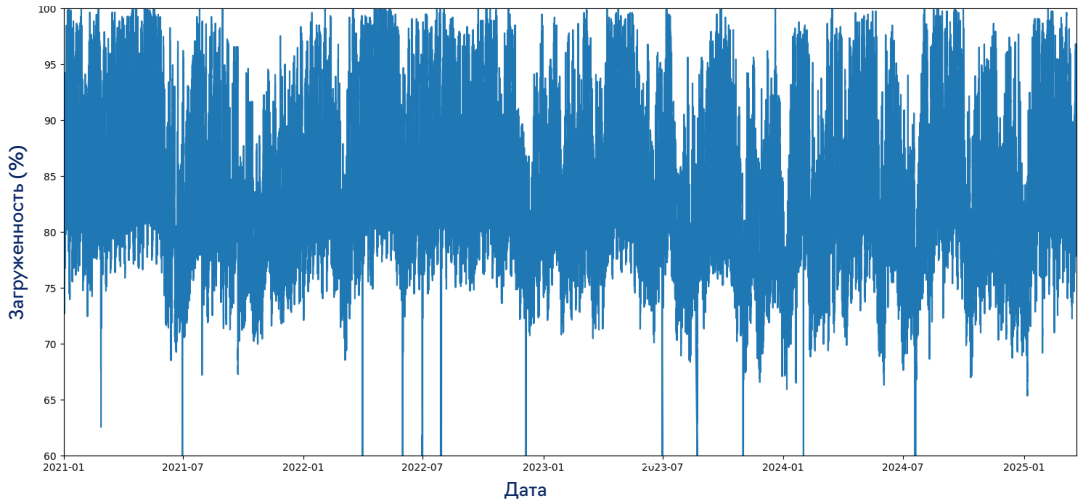
Требования к реализации:

- Покрытие тестами не менее 90%.
- Автоматические развертывание посредством Docker.
- Возможность внешней конфигурации сервиса во время работы.
- Логирование всех модулей сервиса.

Анализ данных реальной загрузки

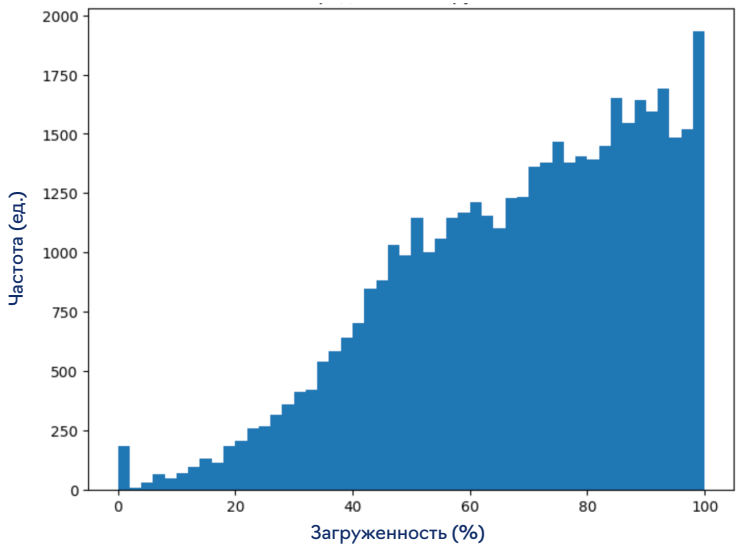
Для анализа были использованы данные с 2021 по 2024 год

Загруженность GPU суперкомпьютера

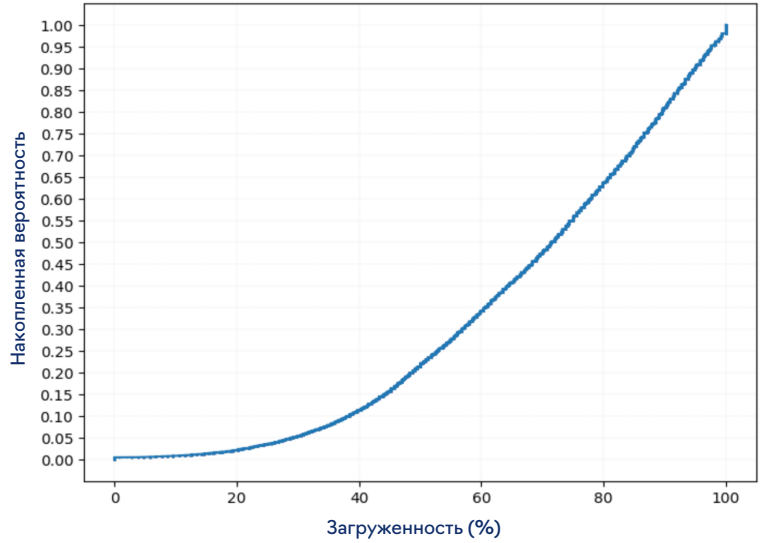


Поиск аномалий

Распределение загруженности GPU суперкомпьютера



Функция распределения загруженности GPU суперкомпьютера



Поиск сезонностей

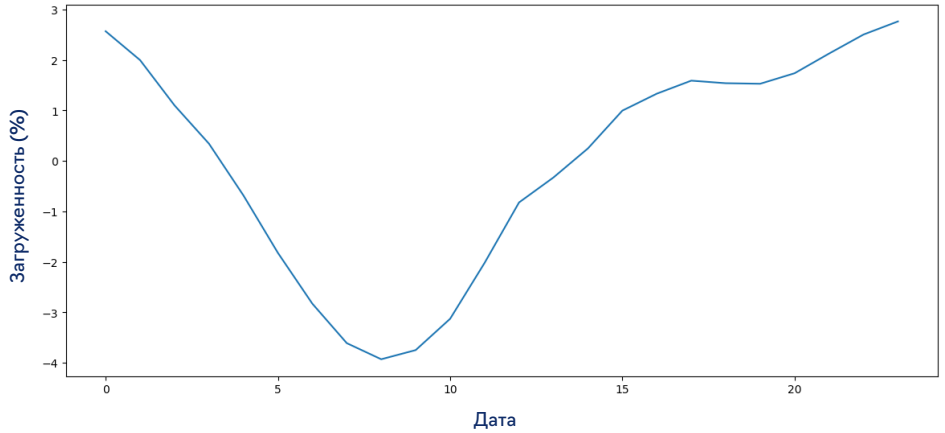
В данных присутствуют три ярко выраженные сезонности: **дневная, недельная и годовая**.

Из графика дневной сезонности можно сделать вывод, что наименьшая загруженность кластера в период с 5 ночи до 10 утра.

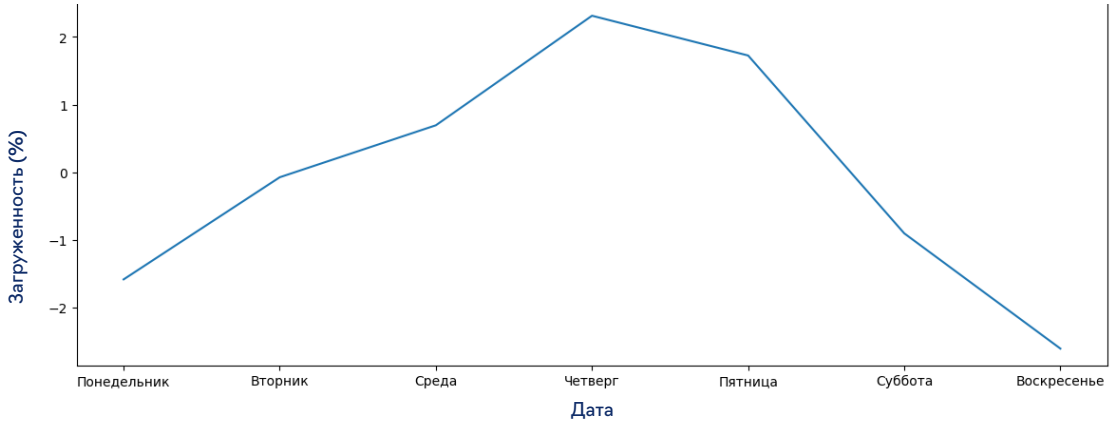
Также видно, что загруженность в выходные снижается.

Кроме этого, можно определить, что самые “свободные” месяцы – июль и ноябрь.

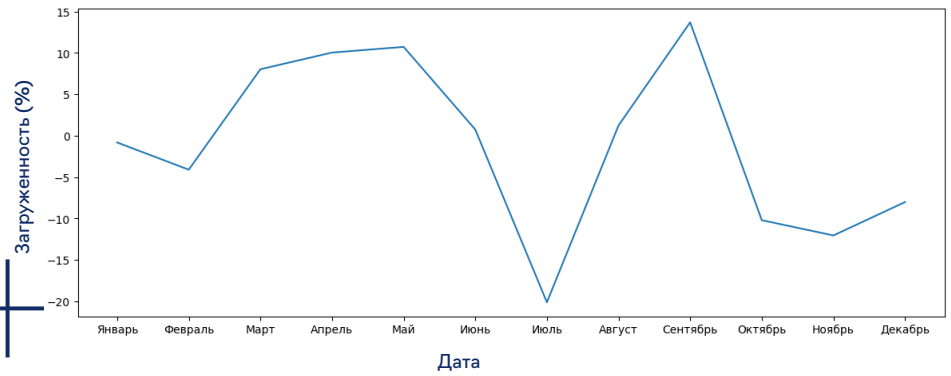
Дневная сезонность



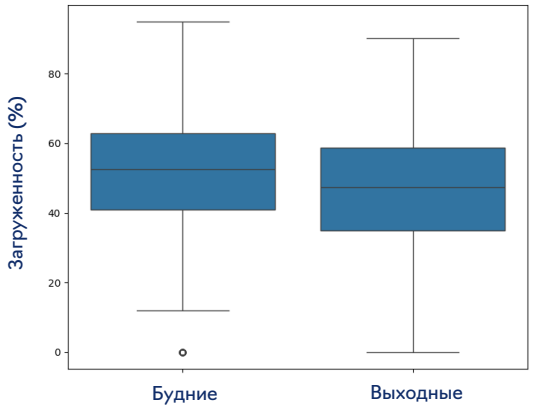
Недельная сезонность



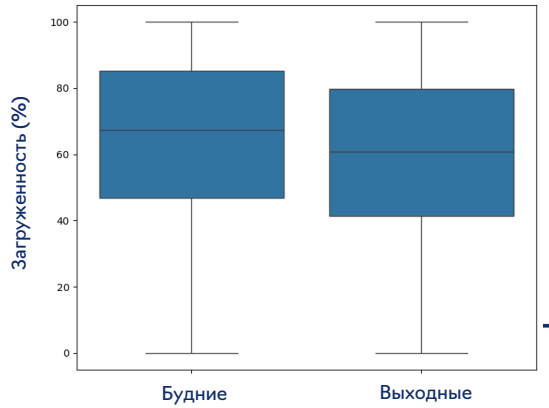
Годовая сезонность



Сравнение GPU загрузки: Будние vs Выходные

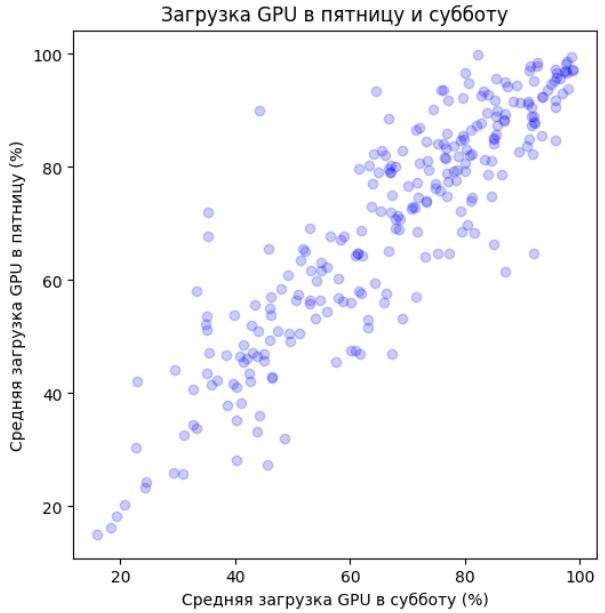


Сравнение CPU загрузки: Будние vs Выходные



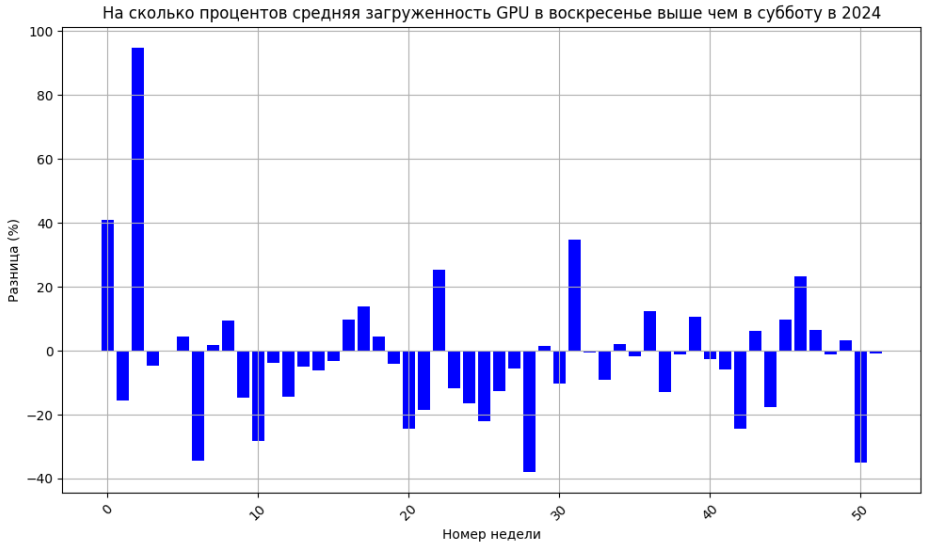
Анализ сезонностей

Между будними и выходными, в частности между пятницей и субботой видна значимая корреляция



-7.4% разница медиан загруженности CPU между будними и выходными

-9.3% разница медиан загруженности GPU между будними и выходными



В воскресенье в 48% случаев нагрузка была выше, чем в субботу

Вывод: необходимо прогнозировать оба выходных дня.

Для прогнозирования можно использовать предшествующие значения.

Способы запуска дополнительных вычислительных задач на суперкомпьютере

Воздействия на планировщик задач SLURM

Изменение квот пользователей

Минусы:

- Нельзя задать длительность задач. Может запуститься задача на месяц вперед.
- Нельзя заранее определить, как именно нужно изменить квоты.

Изменение приоритетов конкретных задач

Минусы:

- Даже с повышенным приоритетом задача все равно будет стоять в очереди SLURM.
- Задача с повышенным приоритетом все равно может не влезать в квоты пользователя.

Что если не изменять приоритеты и квоты, а запускать отдельные задачи

Запуск дополнительных задач в обход планировщика SLURM

- Анализирует текущие и будущие свободные ресурсы
- Анализирует максимальное время выполнения задачи
- Анализирует теги задач

Прогнозирование загруженности суперкомпьютера

Так как постоянно изменяется модель поведения пользователей, проводится оптимизация работы и конфигурации кластера, использование математических методов затруднено, поэтому было принято решение использовать методы машинного обучения.

Одним из важных факторов при подборе модели – **интерпретируемость**, чтобы не позволять системе принимать непредсказуемые решения.

Выбранные модели предсказания временных рядов:

- SARIMAX
- TBATS
- Prophet
- NeuralProphet

Модели сравниваются по **Mean Segmented Squared Error**

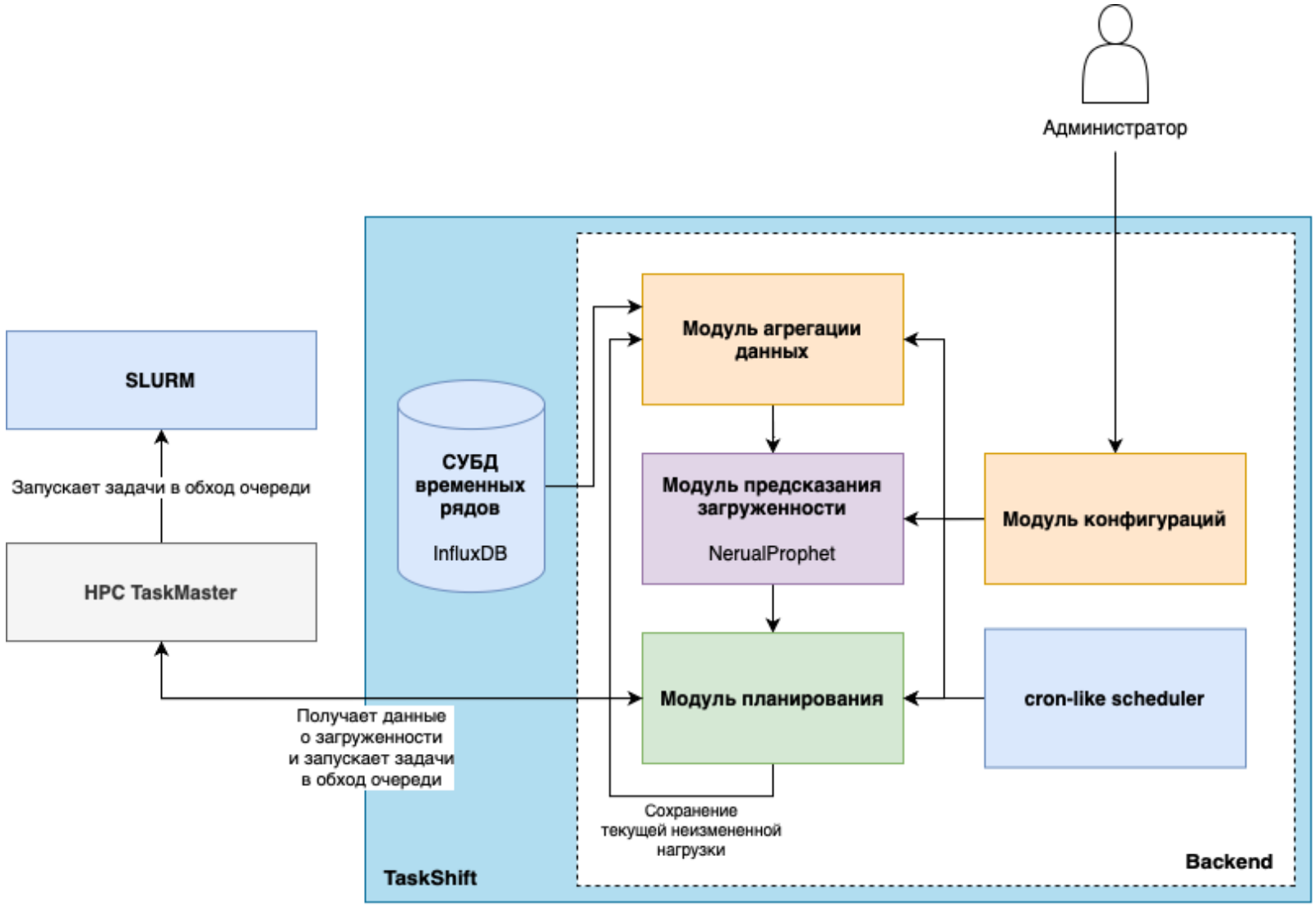
$$\mathrm{MSSE} = \frac{1}{K}\sum_{k=1}^{K}\left(\left(\bar{y}_k - \bar{\hat{y}}_k\right)^2\right), \quad \text{где} \quad \bar{y}_k = \frac{1}{n_k}\sum_{t\in S_k} y_t, \quad \bar{\hat{y}}_k = \frac{1}{n_k}\sum_{t\in S_k} \hat{y}_t.$$

Учитывает вклад нескольких предсказанных сегментов

Модель	MSSE	Время обучения
NeuralProphet	7.37	4 мин. 24 сек.
Prophet	579.81	3 мин. 17 сек.
SARIMAX	16.71	41 мин. 9 сек.
TBATS	28.77	1 час. 38 мин.

Лучшая модель: **NeuralProphet**

Архитектура сервиса



Модуль агрегирования данных
Выгружает и сохраняет временные ряды

Модуль предсказания загруженности
На основе временных рядов делает предсказания о будущей загруженности

Модуль продвижения
Анализирует загруженность суперкомпьютера с текущей очередью задач и принимает решение о запуске задач в обход планировщика SLURM

Модуль конфигураций
Настраивает работу модулей из заранее созданных конфигураций

Реализация

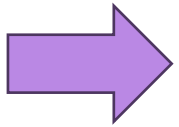
Язык программирования: **Python**

по требованию заказчика

Модуль агрегирования данных

Варианты СУБД для хранения временных рядов:

- QuestDB
- InfluxDB
- Prometheus
- TimescaleDB



InfluxDB

- Уже используется в HPC TaskMaster.
- Понятное API для Python.
- Подробная документация.
- Открытый исходный код.
- Стандарт индустрии.

Загружает данные с помощью pivot запроса

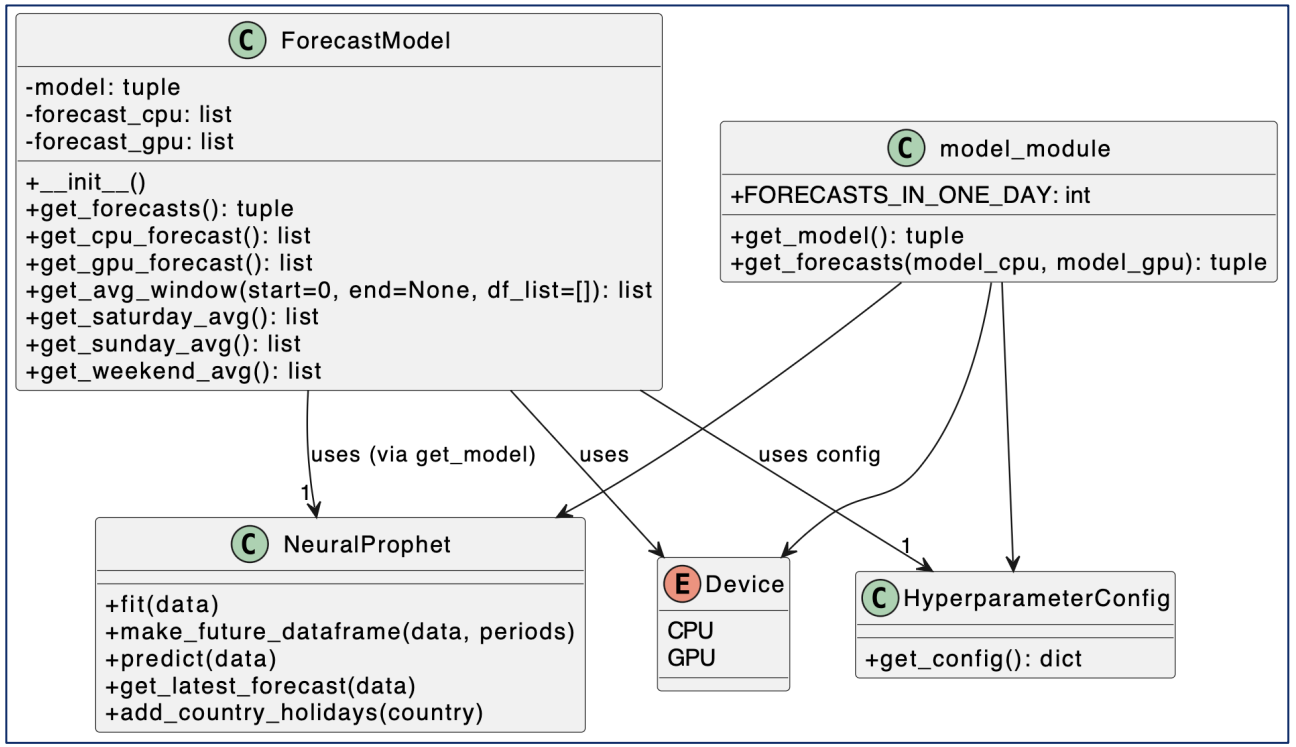
```
query = """from(bucket: "cluster_load")
|> range(start: 1970-01-01T00:00:00Z)
|> filter(fn: (r) => r._measurement == "gpu_load" or r._measurement
   == "cpu_load")
|> pivot(
    rowKey: ["_time"],
    columnKey: ["_measurement"],
    valueColumn: "_value"
)
"""
```

Модуль предсказания временных рядов

Получает данные из модуля агрегации и конвертирует их в пригодный для модели формат

Для предсказаний используется **NeuralProphet**

Предоставляет интерфейс для доступа к предсказаниям модели

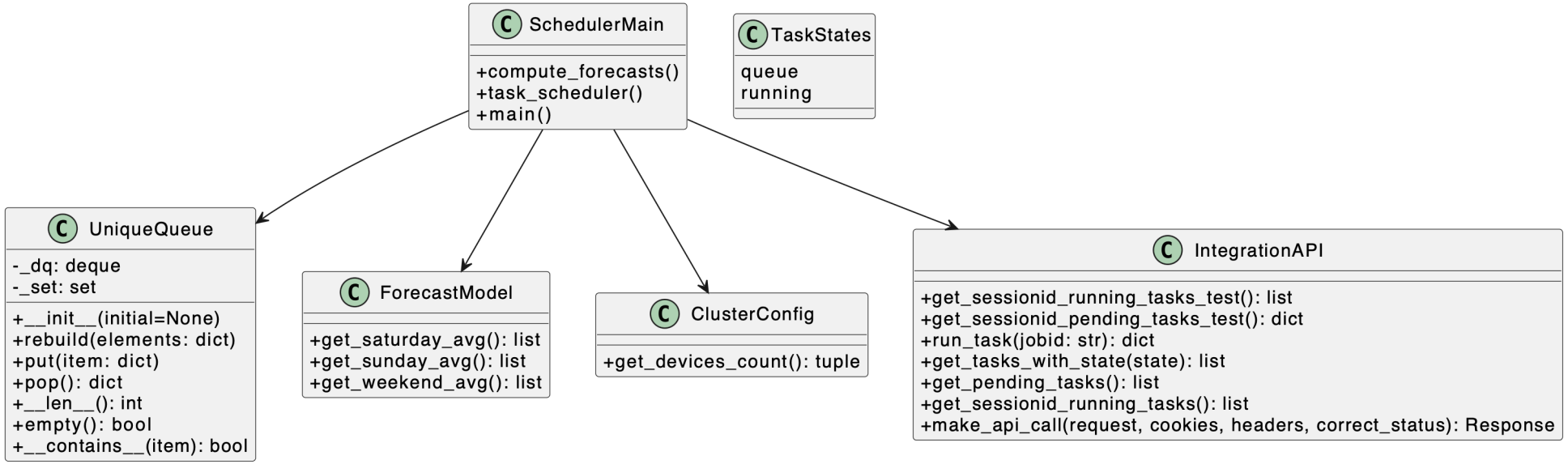


Модуль продвижения

- Для запуска системы каждую пятницу использует APscheduler с cron-like таймерами.
- Запрашивает информацию о состоянии очереди каждую минуту.
- Сохраняет текущие данные о загруженности системы без учета запущенных им задач, чтобы не загрязнять данные.

- Использует быструю имплементацию очереди, которая сохраняет порядок с каждой перестройкой.

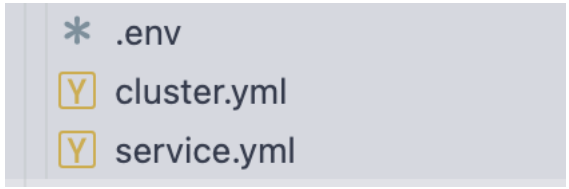
`rebuild`	$\mathcal{O}(n+m)$
`put`	$\mathcal{O}(1)$
`pop`	$\mathcal{O}(1)$



Модуль конфигурации

Используется для определения конфигурации кластера, настройке гиперпараметров и ключей доступа.

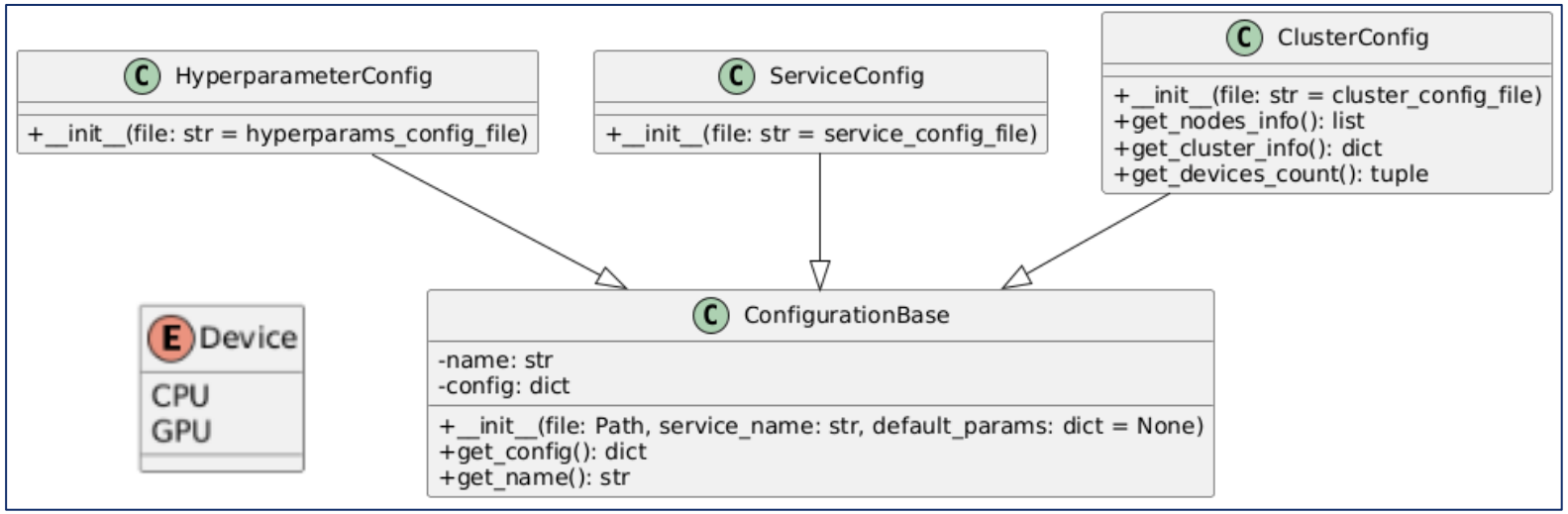
Для конфигурации используются **.yml** и **.env** файлы



REST API реализован с помощью **FastAPI**

```
PATCH /scheduler/config/setmaxload/ HTTP/1.1
Host: localhost
Content-Type: application/json
Accept: application/json
{
  "max_cpu_load": 85.5,
  "max_gpu_load": 90.0
}
```

Для инкапсуляции доступа к параметрам конфигурации для каждого модуля используются отдельные классы



Результаты

В результате была достигнута поставленная цель: **разработана система динамического изменения приоритетов,** благодаря проведенному анализу данных и предложенной архитектуре.

Проект использует внешнюю СУБД **InfluxDB v2** для хранения данных, прогнозирует временной ряд с помощью модели предсказания **NeuralProphet** и анализирует текущий пул задач с помощью разработанного модуля продвижения.

Кроме этого выполнены остальные задачи:

1. Код сервиса покрыт юнит-тестами на ~98% в среднем
2. Развертывание сервиса автоматизировано с помощью Docker
3. Логирование работы модулей
4. Сделан внешний API для взаимодействия с сервисом

Загруженность суперкомпьютера на тестовых данных выросла на 8%

Система в симуляторе запустила 12 задач сверх запланированного, тем самым оптимизировав загрузку суперкомпьютера.

Апробация

Работа была апробирована на всероссийской научной конференции “Параллельные вычислительные технологии 2025”, прошедшей в Москве с 8 по 10 апреля 2025 г.

По результатам работы опубликована научная статья РИНЦ.

Список литературы

[1] A beginner's guide to SLURM. url: https://github.com/influxdata/influxdb-client-python.
[2] APScheduler documentation. url: https://apscheduler.readthedocs.io/en/3.x/.
[3] Thierry Blu, Philippe Th´evenaz и Michael Unser. "Linear interpolation revitalized". В: IEEE Transactions on Image Processing 13.5 (2004), с. 710—719.
[4] Inc Docker и др. "Docker". В: Inea].[Junio de 2017]. Disponible en: https://www. docker. com/what-docker (2020).
[5] DotEnv GitHub repository. url: https://github.com/theskumar/python-dotenv.
[6] FastAPI documentation. url: https://fastapi.tiangolo.com/.
[7] influxDB python client. url: https : / / github . com / influxdata / influxdb - client - python.
[8] Pavel Kostenetskiy, Roman Chulkevich и Viachesvlav Kozyrev. "HPC Resources of the Higher School of Economics". В: Journal of Physics: Conference Series 1740.1 (янв. 2021), с. 012050. doi: 10.1088/1742- 6596/1740/1/012050. url: https://dx.doi.org/10.1088/1742-6596/1740/1/012050.
[9] Pavel Kostenetskiy, Artemiy Shamsutdinov, Roman Chulkevich, Vyacheslav Kozyrev и Dmitriy Antonov. "HPC TaskMaster–Task Efficiency Monitoring System for the Supercomputer Center". В: International Conference on Parallel Computational Technologies. Springer. 2022, с. 17—29.
[10] likeC4 framework official site. url: https://likec4.dev/.
[11] Loguru GitHub repository. url: https://github.com/Delgan/loguru.
[12] Hoang Minh Nguyen, Gaurav Kalra и Daeyoung Kim. "Host load prediction in cloud computing using long short-term memory encoder–decoder". В: The Journal of Supercomputing 75.11 (2019), с. 7592—7605.
[13] Brian Okken. Python Testing with pytest. Pragmatic Bookshelf, 2022.
[14] SLURM Workload Manager. url: https://slurm.schedmd.com.
[15] Oskar Triebe, Hansika Hewamalage, Polina Pilyugina, Nikolay Laptev, Christoph Bergmeir и Ram Rajagopal. "Neuralprophet: Explainable forecasting at scale". В: arXiv preprint arXiv:2111.15397 (2021).
35
[16] Oskar Triebe, Nikolay Laptev и Ram Rajagopal. "Ar-net: A simple auto-regressive neural network for time-series". В: arXiv preprint arXiv:1911.12436 (2019).
[17] Supercomputer Modeling Unit. HPC TaskMaster repository GitLab. url: https://git.hpc.hse.ru/open-source/hpc-taskmaster (дата обр. 03.02.2025).
[18] Supercomputer Modeling Unit. HSE University HPC Cluster cHARISMa. url: https : //hpc.hse.ru/hardware/hpc-cluster (дата обр. 27.01.2025).
[19] Andrea V´azquez-Ingelmo, Alicia Garc´ıa-Holgado и Francisco J Garc´ıa-Pe˜nalvo. "C4 model
in a software engineering subject to ease the comprehension of uml and the software". В: 2020 IEEE Global Engineering Education Conference (EDUCON). IEEE. 2020, с. 919—924.
[20] Vadim Voevodin, Roman Chulkevich, Pavel Kostenetskiy, Vyacheslav Kozyrev, Anton Maliutin,
Dmitry Nikitenko, Sergey Rykovanov, Artemiy Shamsutdinov, Yuriy Shkandybin и Sergey Zhumatiy. "Administration, Monitoring and Analysis of Supercomputers in Russia: a Survey of 10 HPC Centers". В: Supercomputing Frontiers and Innovations 8.3 (окт. 2021), с. 82—103. doi: 10.14529/jsfi210305. url: https://superfri.org/index.php/superfri/article/view/397.
[21] Leland Wilkinson. "The grammar of graphics". В: Handbook of computational statistics: Concepts and methods. Springer, 2011, с. 375—414.
[22] Сергей Анатольевич Жуматий и Константин Сергеевич Стефанов. Суперкомпьютеры:
администрирование. ООО «МАКС Пресс», 2018.
[23] П.С. Костенецкий, А.Б. Шамсутдинов, Р.А. Чулкевич и В.И. Козырев. "Разработка подсистемы анализа эффективности использования вычислительных ресурсов для системы HPC TaskMaster". В: Параллельные вычислительные технологии (ПаВТ'2023). 2023, с. 155—161