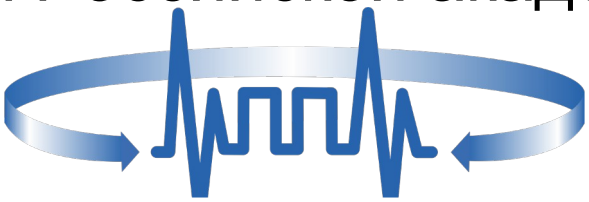
Автоматизация обновления инструментария SvF на платформе Everest с использованием Docker

Сергей Смирнов
Владимир Волошинов

Центр распределённых вычислений
Института проблем передачи информации им. А.А. Харкевича Российской академии наук



Исследование выполнено за счет гранта Российского научного фонда № 25-11-00349,
https://rscf.ru/project/25-11-00349

Платформа Эверест

Everest – PaaS для создания REST-сервисов из приложений командной строки

Для пользователя:

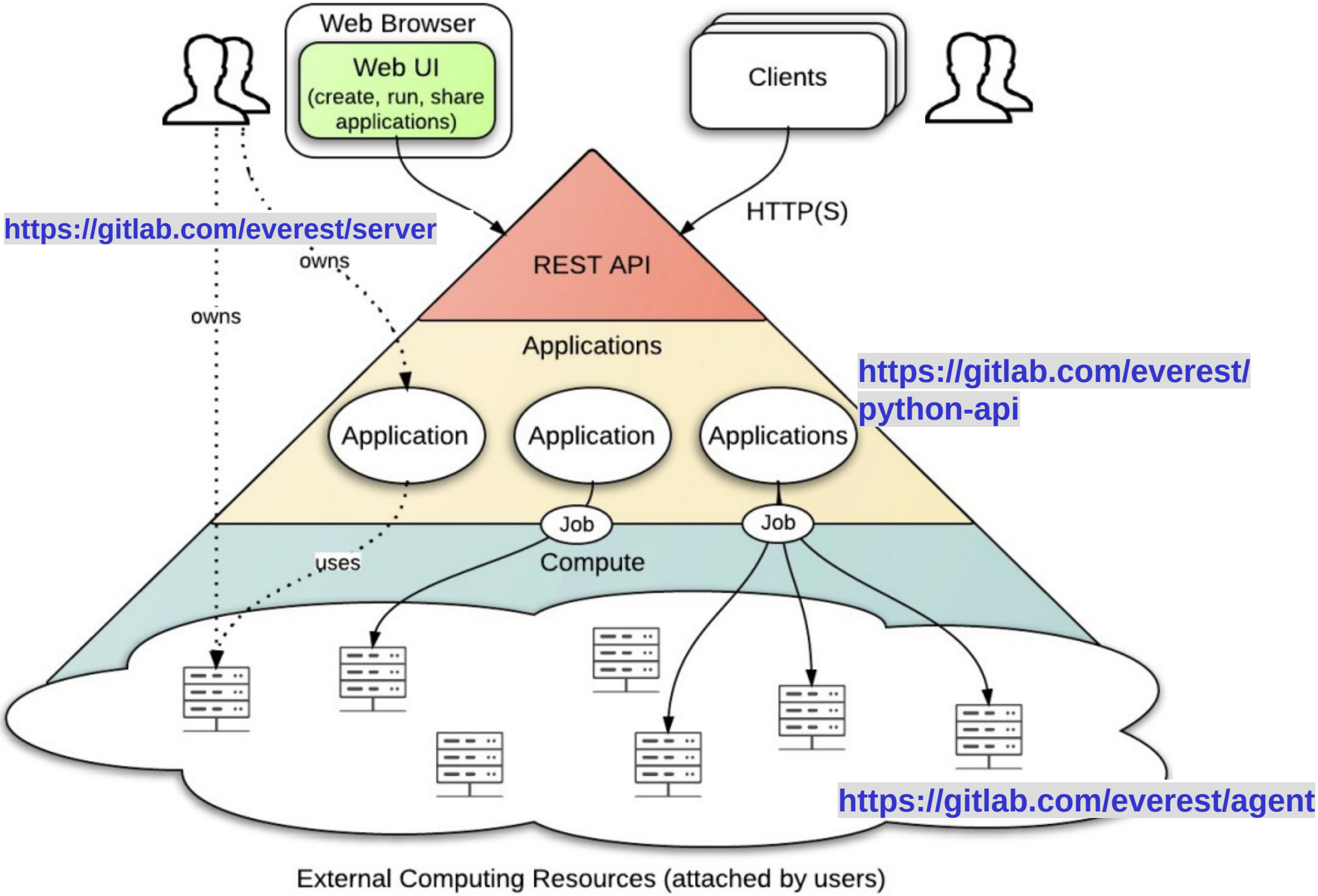
- Использование приложения через привычную Веб-форму
- Отслеживание статуса расчета
- Проведение множества расчетов одновременно

Для разработчика:

- Простая публикаця приложения: настроить параметры приложения через Веб-интерфейс и запустить агента (можно на собственном лэптопе)
- Готовый API
- Готовая авторизация пользователей

Платформа Everest, everest.distcomp.org

Позволяет быстро создавать REST-сервисы для проведения расчетов



Развертывание вычислительного ресурса

1. Подготовка окружения:
   - Установить компиляторы, интерпретаторы
   - Собрать солверы и другие пакеты
   - Установить агент Эвереста (скачать, создать токен, отредактировать конфиг, настроить автозагрузку)
2. Варианты получения кода приложения:
   - Вручную скопировать в заранее известную папку
   - **Скачивать при каждом запуске задания**
   - Загрузить в Эверест как один из файлов приложения

Окружение меняется редко, настраивается долго

Приложение меняется часто, копируется быстро

Контейнеризация

Контейнеризация – это метод виртуализации, при котором приложение и все его зависимости упаковываются в изолированный контейнер. Основные характеристики:

- Контейнер включает все необходимое для работы приложения (библиотеки, конфигурации, файлы)
- Каждый контейнер изолирован от операционной системы и других контейнеров
- Контейнер можно легко переносить между разными системами
- Быстрый запуск - секунды вместо минут (в отличие от виртуальных машин)
- Эффективное использование ресурсов - контейнеры используют общее ядро операционной системы

SvF-технология

SvF (Simplicity vs Fitting) – технология для поддержки построения математических моделей (структурных, не только регрессионных) на основе экспериментальных данных.

Вычислительная схема SvF основана на решении двухуровневых оптимизационных задач специального типа. На нижнем уровне необходимо решить набор независимых оптимизационных задач, аналогичных обратным задачам с регуляризацией Тихонова.

Зависимости:

- Python 3.10+ (Pyomo, Everest Python API, ...)
- Солвер Ipopt

Задания и подзадания SvF-расчета в Everest

jobs 〉 svf-remote-3-Oscill_K_mu

Job Info | Share | Tasks | Children

done Task-0

shark1vvv4SvF 2m 27s Wait 0ms Run 2m 25s 0.63 KB 115.50 KB

jobs 〉 svf-remote-3-Oscill_K_mu

Job Info | Share | Tasks | Children

Name	Application	State	Submitted
NoName0-ipopt-1	solve-set-opt-probs	DONE	18 Nov 2024 00:18:14
NoName1-ipopt-1	solve-set-opt-probs	DONE	18 Nov 2024 00:18:33
NoName1-ipopt-1	solve-set-opt-probs	DONE	18 Nov 2024 00:18:53
NoName1-ipopt-1	solve-set-opt-probs	DONE	18 Nov 2024 00:19:13
NoName1-ipopt-1	solve-set-opt-probs	DONE	18 Nov 2024 00:19:33
NoName2-ipopt-1	solve-set-opt-probs	DONE	18 Nov 2024 00:19:53
NoName3-ipopt-1	solve-set-opt-probs	DONE	18 Nov 2024 00:20:13

jobs 〉 NoName3-ipopt-1

Job Info | Share | Tasks

done Task-0 shark1vvv 6s Wait 0ms Run 5s 42.88 KB 6.16 KB

done Task-1 shark1vvv 6s Wait 0ms Run 5s 42.88 KB 6.18 KB

done Task-2 shark1vvv 6s Wait 0ms Run 5s 42.88 KB 6.16 KB

done Task-3 shark1vvv 6s Wait 0ms Run 5s 42.88 KB 6.12 KB

done Task-4 shark1vvv 6s Wait 0ms Run 5s 42.88 KB 6.18 KB

done Task-5 shark1vvv 6s Wait 0ms Run 5s 42.88 KB 6.15 KB

done Task-6 shark1vvv 6s Wait 0ms Run 5s 42.88 KB 6.15 KB

done Task-7 shark1vvv 6s Wait 0ms Run 5s 42.88 KB 6.16 KB

done Task-8 shark1vvv 6s Wait 0ms Run 5s 42.88 KB 6.12 KB

done Task-9 shark1vvv 6s Wait 0ms Run 5s 42.88 KB 6.17 KB

shark1vvv 6s Wait 0ms Run 5s 42.88 KB 6.17 KB

Идея

Два уровня контейнеров (агент + SvF+солверы), схема версионирования и роли компонентов.

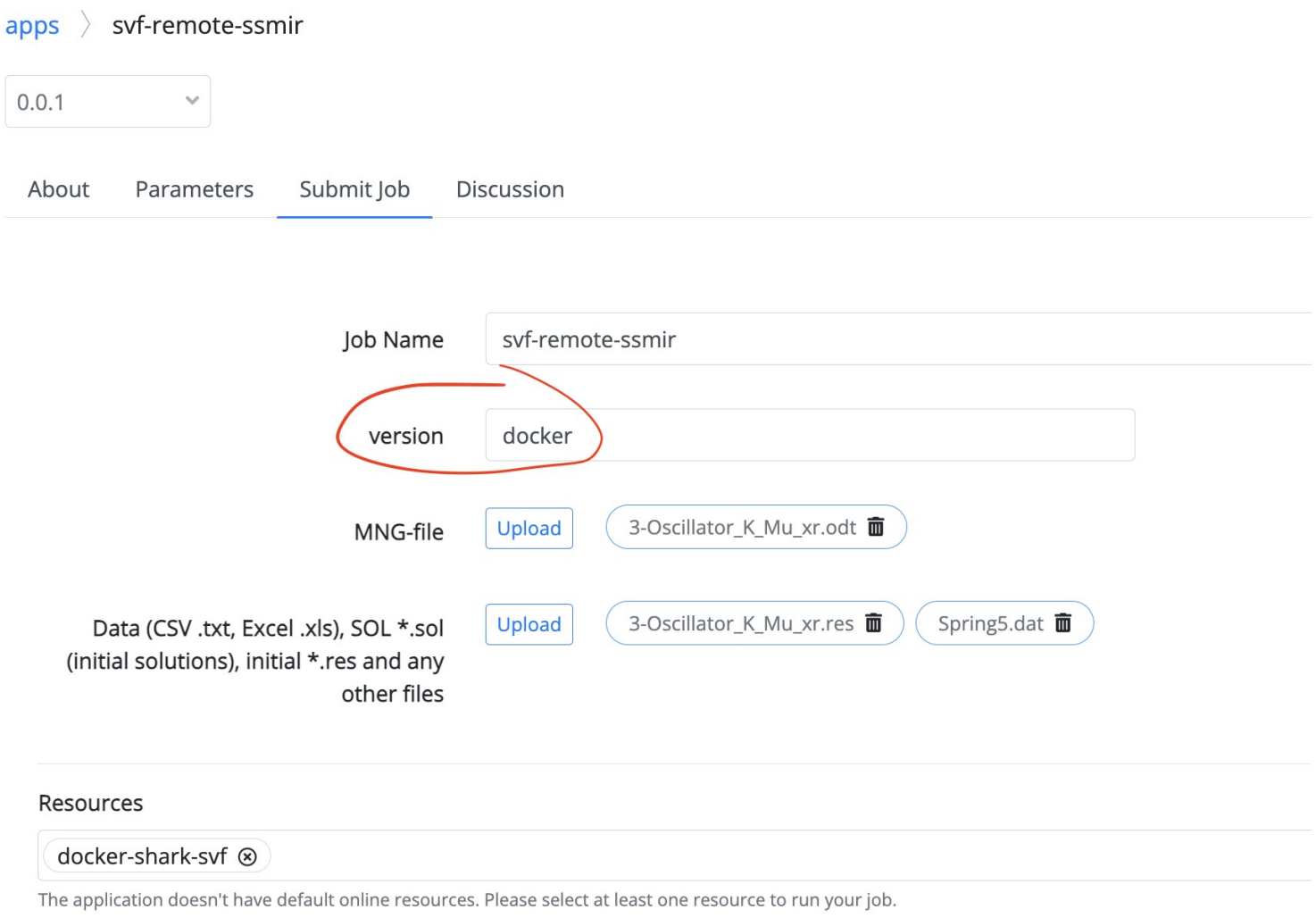
Реализация: CI/CD через GitHub Actions, схема тегов (<branch>, cached-...), бутстрап-скрипты и их поведение для разных тегов.

Интеграция: параметр версии в приложениях Everest, версия в план-файлах и запуск агентского контейнера с монтированием Docker-сокета.

Передача версии

Задаем версию SvF через тег докер образа в репозитории

https://github.com/distcomp/SvF/pkgs/container/svf



Настройка агента

Внутри установлены:

- Docker CLI (`docker`);
- скрипты `runSvF-remote.sh`, `run-ipopt.sh`, `run-scip.sh`.

Запуск:

```
    docker run -d --restart always --name svf-agent \
--hostname $(hostname)-svf \
-e EVEREST_TOKEN=!!!TOKEN!!! -e MAX_TASKS=16 \
-v /var/run/docker.sock:/var/run/docker.sock \
ghcr.io/distcomp/svf-agent:latest
```

Агент подключается к Everest, а все задачи SvF/солверов запускаются уже как дочерние контейнеры SvF+солверов через Docker внутри агента.

Результаты

Централизованное управление версиями SvF и солверов через Docker-образы

Автообновление при `git push` без ручных действий на вычислительных узлах.

Возможность легко переключаться между версиями через параметр версии

Спасибо за внимание!

Сергей Смирнов,
ИППИ им. Харкевича РАН
sasmir@gmail.com