Balanced identification of mathematical models as a service of the Everest distributed computing platform

Vladimir Voloshinov, Alexander Sokolov
http://bit.ly/VoloshinovGScholar

Center for Distributed Computing,
Kharkevich Institute for Information Transmission Problems,

This work was supported by the Russian Science Foundation under grant no. 22-11-00317, https://rscf.ru/project/22-11-00317/

SvF-technology

SvF (Simplicity vs Fitting) – a technology to support construction of mathematical models (structural, not regression only) based on experimental data.

It has been successfully applied in biology, population dynamics, plant physiology, plasma physics, meteorology, ecology… Freely available in source code **https://github.com/distcomp/SvF**

Basic publications (A. Sokolov had used this method repeatedly several years before)

1. Sokolov A., Voloshinov V. **Model Selection by Balanced Identification: the Interplay of Optimization and Distributed Computing** // Open Computer Science, **2020**

- Соколов А.В., Волошинов В.В. **Выбор математической модели: баланс между сложностью и близостью к измерениям** // International Journal of Open Information Technologies, **2018**

- find other references here **https://github.com/distcomp/SvF#references**

- up-to-date User Manual **https://github.com/.../SvF/SvF_UserGuide.pdf**

The SvF computational scheme is based on solving special-type bi-level optimization problems. Here, at the lower level, it is necessary to solve a set of independent optimization problems similar to the inverse problems with the Tikhonov regularization.

General flow-chart of SvF-technology

The main problem: to construct and to validate mathematical model of a *phenomenon* by a set of experimental data

Inputs: Experimental data (relational DB tables *.txt, *.xls … see User Manual)

Hypothesis about the structure of mathematical model and its formulation (as a set of equations, inequalities, may be integro-differential expressions) with explicit indications which parameters, or functions need to be determined (be identified).

All other steps may be performed automatically:

1. Formulation of bilevel optimization problem
2. Automatic discretization of optimization problems, if it is required, e.g. if differential and/or integral equations are presented in the model.
3. Automatic generation of SvF-computing scenario as a Python program
4. Implementation of SvF-scenario either locally, or in Everest distributed computing environment, http://everest.distcomp.org/

Example. “Inverse problem” for ODE

Identify a model of dynamic system by a set of trajectory points known with error.

Hypothesis: ODE (of order 1, 2). We need **to identify right side of ODE**.

$$\left.\begin{array}{l} \boxed{\{(\tilde{x}_d, t_d) : d=1:N_d\}, \tilde{x}_d = x(t_d) \pm \text{err}} \\ \dot{x} = F(t, x(t)) \mid \ddot{x} = F(t, x(t)),\ t \in [t_{\mathbf{L}}, t_{\mathbf{U}}] \end{array}\right\} \Rightarrow \boldsymbol{x(t), F(t,x) - ?}$$

SvF lower-level problem: **cross-validation sub-problems**

Example. “Inverse problem” for ODE

Identify a model of dynamic system by a set of trajectory points known with error.

Hypothesis: ODE (of order 1, 2). We need **to identify right side of ODE**.

$$\left.\begin{array}{l}\boxed{\{(\tilde{x}_d, t_d) : d=1:N_d\}, \tilde{x}_d = x(t_d) \pm \text{err}} \\ \dot{x} = F(t, x(t)) \mid \ddot{x} = F(t, x(t)),\ t \in [t_\mathbf{L}, t_\mathbf{U}]\end{array}\right\} \Rightarrow \boldsymbol{x(t), F(t,x)} - ?$$

SvF CV lower-level problems: **“training data-set”**

$$\boldsymbol{D_k} \doteq \{d=1:N_d, d \neq k\}, k=1:N_d$$

regularization coefficients

$$\boldsymbol{\alpha} = (\alpha_t, \alpha_x)$$

$$\underbrace{\frac{1}{|\boldsymbol{D_k}|} \sum_{d \in \boldsymbol{D_k}} (\tilde{x}_d - x(t_d))^2}_{\text{data fitting}} + \underbrace{R(\boldsymbol{\alpha}, F(\cdot,\cdot))}_{\text{simplicity (smooth.)}} \to \min_{x(\cdot), F(\cdot,\cdot)},$$

$$\dot{x} = F(x(t)) \mid \ddot{x} = F(x(t)),\ t \in [t_\mathbf{L}, t_\mathbf{U}],$$

$$x(\cdot) \in C^2[t_\mathbf{L}, t_\mathbf{U}], F(\cdot) \in C^2[x_\mathbf{L}, x_\mathbf{U}].$$

$$R(\boldsymbol{\alpha}, F(\cdot,\cdot)) =$$

$$\int_{t_\mathbf{L}}^{t_\mathbf{U}} \int_{x_\mathbf{L}}^{x_\mathbf{U}} \left(\alpha_t^2 (F''_{tt}(t,x))^2 + 2\alpha_t\alpha_x (F''_{tx}(t,x))^2 + \alpha_x^2 (F''_{xx}(t,x))^2\right) dt\,dx$$

SvF lower-level problem: **cross-validation sub-problems** $\mathcal{P}(\boldsymbol{D_k}, \boldsymbol{\alpha}) \Rightarrow x_{\boldsymbol{k}}^{\boldsymbol{\alpha}}(\cdot)$

Example. Inverse problem for ODE. Solving bilevel opt. prob.

CV-error for a fixed $\boldsymbol{\alpha} = (\alpha_t, \alpha_x)$ $\quad \mathcal{P}(\boldsymbol{D_k}, \boldsymbol{\alpha}) \Rightarrow x_{\boldsymbol{k}}^{\boldsymbol{\alpha}}(\cdot) \quad (k{=}1{:}N_d)$

$$\boldsymbol{\sigma}_{\textbf{cv}}(\boldsymbol{\alpha}) = \sqrt{\frac{1}{N_d} \sum_{k=1:N_d} \left(\tilde{x}_k - x_{\boldsymbol{k}}^{\boldsymbol{\alpha}}(t_{\boldsymbol{k}})\right)^2}$$

SvF upper-level problem: **minimize CV-error by** $\boldsymbol{\alpha}$

$$\left\{\mathcal{P}(\boldsymbol{D_1}, \boldsymbol{\alpha}) \bullet\bullet\bullet \mathcal{P}(\boldsymbol{D_{N_d}}, \boldsymbol{\alpha})\right\}$$

solving
CV-subproblems

$$\{x_{\boldsymbol{k}}^{\boldsymbol{\alpha}}(\cdot), k{=}1{:}N_d\}$$

updating
$\boldsymbol{\alpha}$

$$\left\{\boldsymbol{\sigma}_{\textbf{cv}}(\boldsymbol{\alpha}) \rightarrow \min_{\boldsymbol{\alpha} \geqslant \mathbf{0}}\right\}$$

optimal $\boldsymbol{\alpha}^*$

$$\mathcal{P}(\boldsymbol{D}, \boldsymbol{\alpha}^*) \Rightarrow \boldsymbol{x}^*(\cdot)$$

$$\boldsymbol{\sigma}_{\textbf{mse}}(\boldsymbol{\alpha}^*) = \sqrt{\frac{\mathbf{1}}{\boldsymbol{N_d}} \sum_{\boldsymbol{k=1:N_d}} \left(\tilde{\boldsymbol{x}}_{\boldsymbol{k}} - \boldsymbol{x}^*(\boldsymbol{t_k})\right)^{\mathbf{2}}}$$

Final result: **Root** Mean **Squared Error**

In current implementation all **variational expressions are discretized** to get **finite dimensional Mathematical Programming Problems** which may be solved by **general purpose optimization solvers (we use Ipopt** and SCIP, optionally**).**

Typical chain of optimization problems

$\boldsymbol{\alpha} = (\alpha_t, \alpha_x) \quad \mathcal{P}(D_k, \boldsymbol{\alpha}) \Rightarrow x_{\boldsymbol{k}}^{\boldsymbol{\alpha}}(\cdot) \quad (k=1{:}N_d)$

$\boldsymbol{\sigma_{\mathsf{cv}}(\alpha) \to \min\limits_{\alpha \geqslant 0}}$

$\boldsymbol{\alpha_0}$

$\boldsymbol{\alpha_1} \leftarrow \boldsymbol{\mathcal{P}(D_1, \alpha_0)} \cdots \boldsymbol{\mathcal{P}(D_{N_d}, \alpha_0)}$

$\boldsymbol{\alpha_1}$

$\boldsymbol{\alpha_2} \leftarrow \boldsymbol{\mathcal{P}(D_1, \alpha_1)} \cdots \boldsymbol{\mathcal{P}(D_{N_d}, \alpha_1)}$

$\boldsymbol{\alpha_2}$

$\cdots \quad \cdots \quad \cdots$

$\boldsymbol{\alpha^*} \leftarrow \boldsymbol{\mathcal{P}(D_1, \alpha_M)} \cdots \boldsymbol{\mathcal{P}(D_{N_d}, \alpha_M)}$

$\boldsymbol{\mathcal{P}(D, \alpha^*) \Rightarrow x^*(\cdot)}$

$$\boldsymbol{\sigma_{\mathsf{mse}}(\alpha^*) = \sqrt{\frac{1}{N_d} \sum_{k=1:N_d} \left(\tilde{x}_k - x^*(t_k)\right)^2}}$$

Why we need remote SvF-service & distributed computing

- SvF-toolkit may be run in local mode when all problems are solved at the same host where SvF-main script is run
- There may be dozens of rather hard optimization problems
- SvF-toolkit already can use Everest to solve a set of independent opt. problems on remote resources in parallel mode
- Even with the aid of Everest the SvF-scenario may be rather time consuming (a few hours) and all this time your comp. must be on.

It would be nice to have an option to run SvF-scenario in a mode "send data and wait notification by mail"…

Maturity of SvF-toolkit enables to do that !

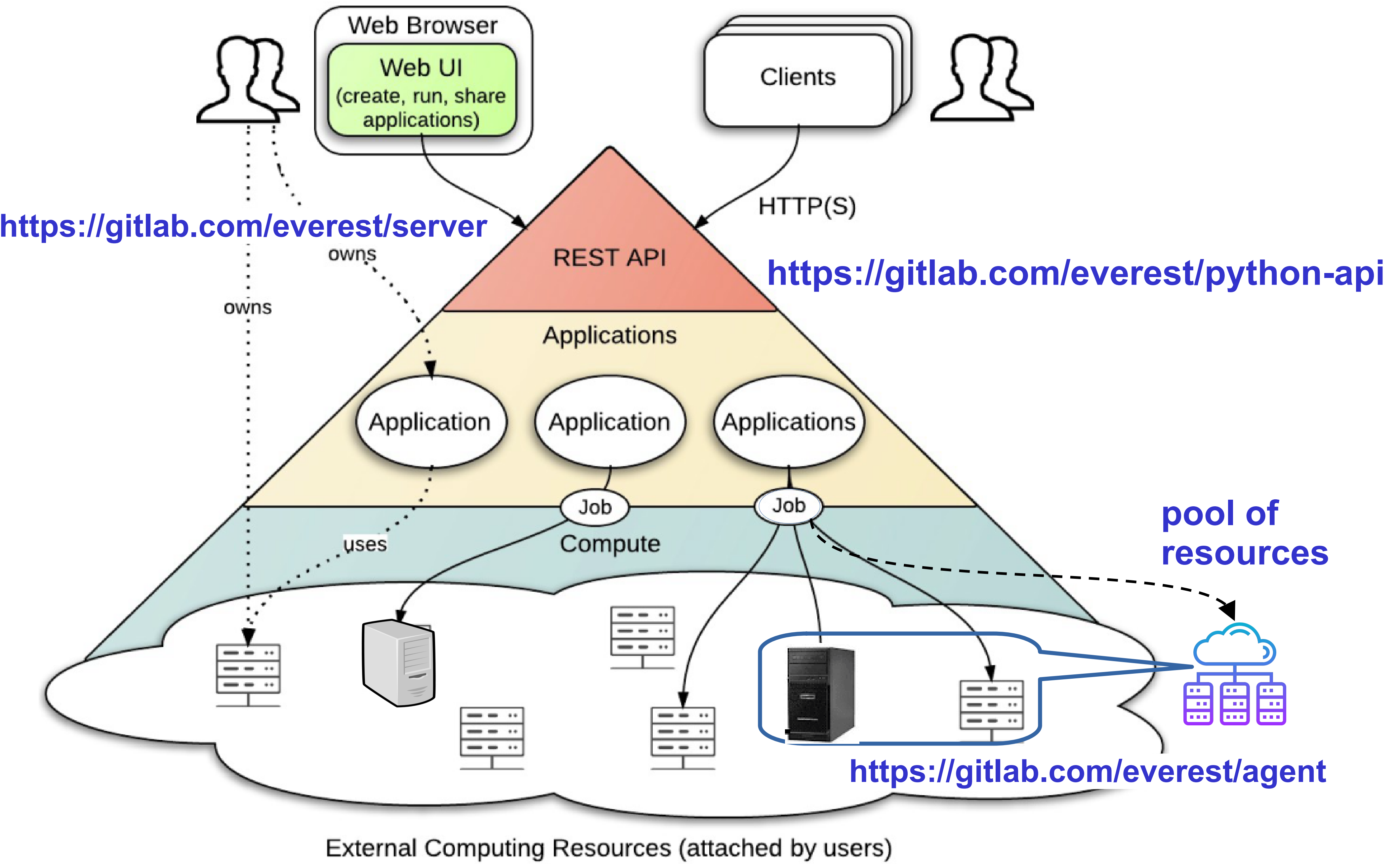
Basic software components of SvF-technology

- Python 3.7+ - basic language
- **Pyomo** toolkit, www.pyomo.org, **PY**thon **O**pt. **M**odeling **O**bjects, Supports all basic types of Math. Programming problems: LP, QP, NLP, MILP, MINLP, Stochastic ..., DAE (Differential and Integral equations), …

  Pyomo supports most of all open source and commercial AMPL-compatible solvers **Ipopt**, **SCIP**, **CBC**, **CPLEX, Gurobi, COPT**, etc
- Everest platform, http://everest.distcomp.org/, and its Optimization Modeling Facet https://optmod.distcomp.org
  - Everest Python API (client side), https://gitlab.com/everest/python-api
  - Everest Application, https://optmod.distcomp.org/apps/vladimirv/SSOP to **s**olve a **s**et of independent **o**ptimization **p**roblems
  - Everest App., https://optmod.distcomp.org/apps/vladimirv/svf-remote to run the SvF-scenario in remote mode

Everest platform architecture outlines

Describe/Develop/Deploy REST-services representing existing applications



svf-remote application in / out parameters

Everest Opt

APPLICATIONS
- List
- Create new

JOBS

RESOURCES
- List
- Create new

GROUPS

ABOUT

DOCUMENTATION

Optimization Modelling Services

Vladimir Voloshinov
vladimirv

apps 〉 svf-remote READY

0.1.0

Unstar ★ 4 | Export | Edit

About | Parameters | Submit Job | Discussion

Inputs

	Title	Name	Type	Values	Default	Description
	Data and *.res	data	array[URI]			
✓	MNG-file	mng	URI			

Input data (txt, xls)

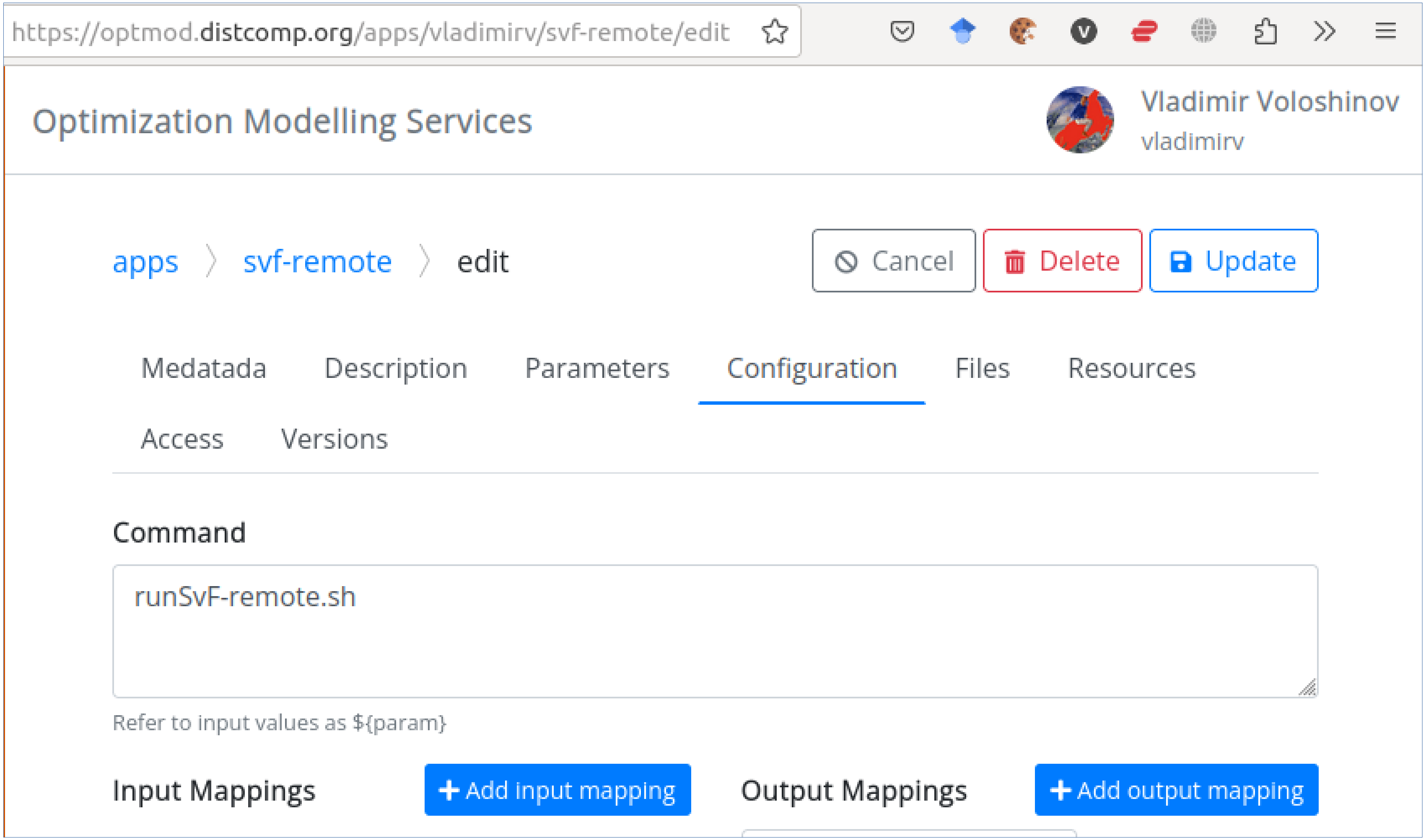
MNG-file (txt, odt) with SvF-task formulation

Outputs

	Title	Name	Type	Description
	results	results	URI	Results *.sol, *.res, *.png
✓	stderr	stderr	URI	Copy of console output
✓	stdout	stdout	URI	Copy of console output

ZIP-arch.

svf-remote implementation is simple (due to SvF features)




```
#!/bin/bash
<PATH TO SvF Folder>/SvF/runSvF31.sh
zip results.zip *.sol *.png *.res
```

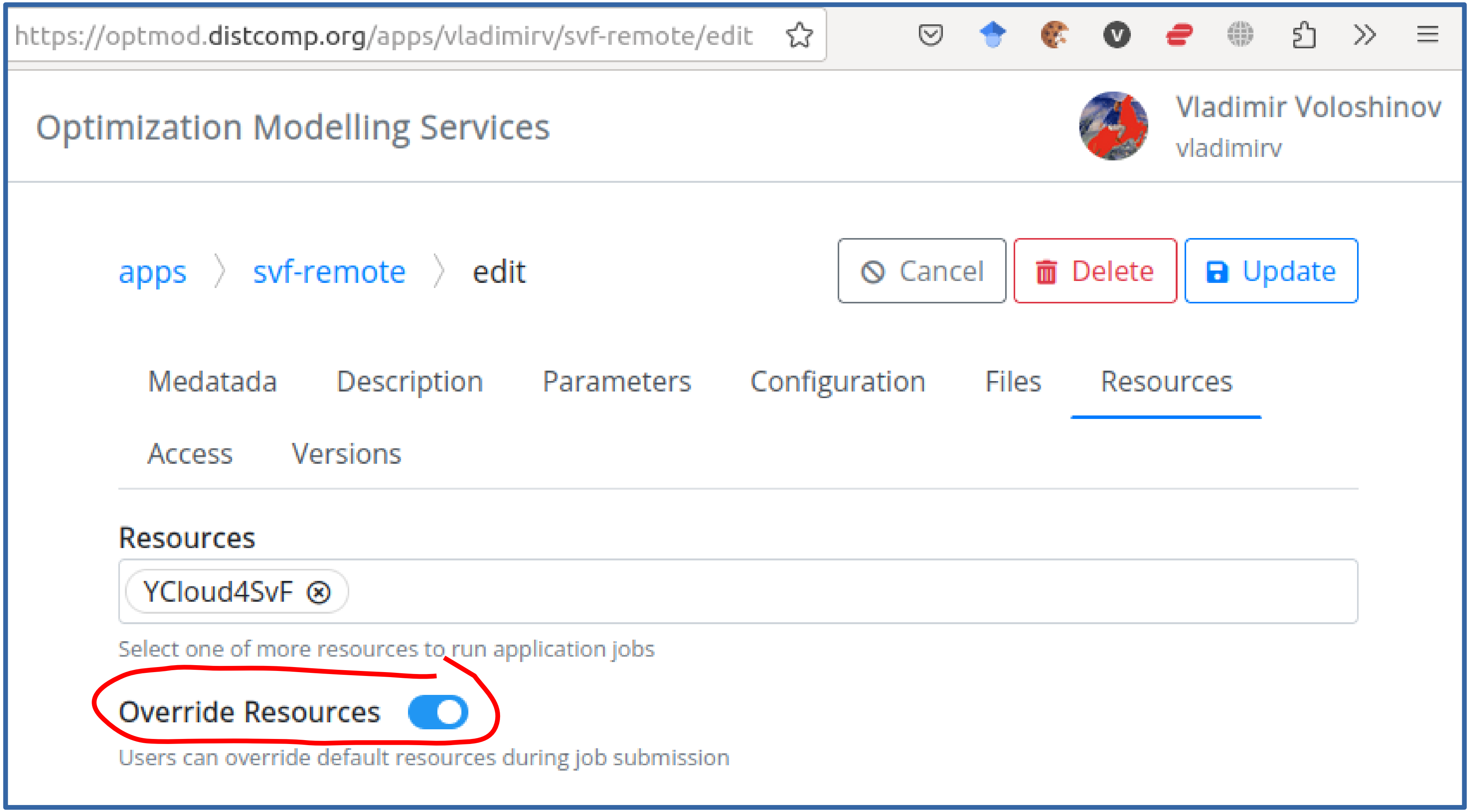
runSvF-remote.sh – somewhere in the $PATH

svf-remote user can choose Everest resource to run

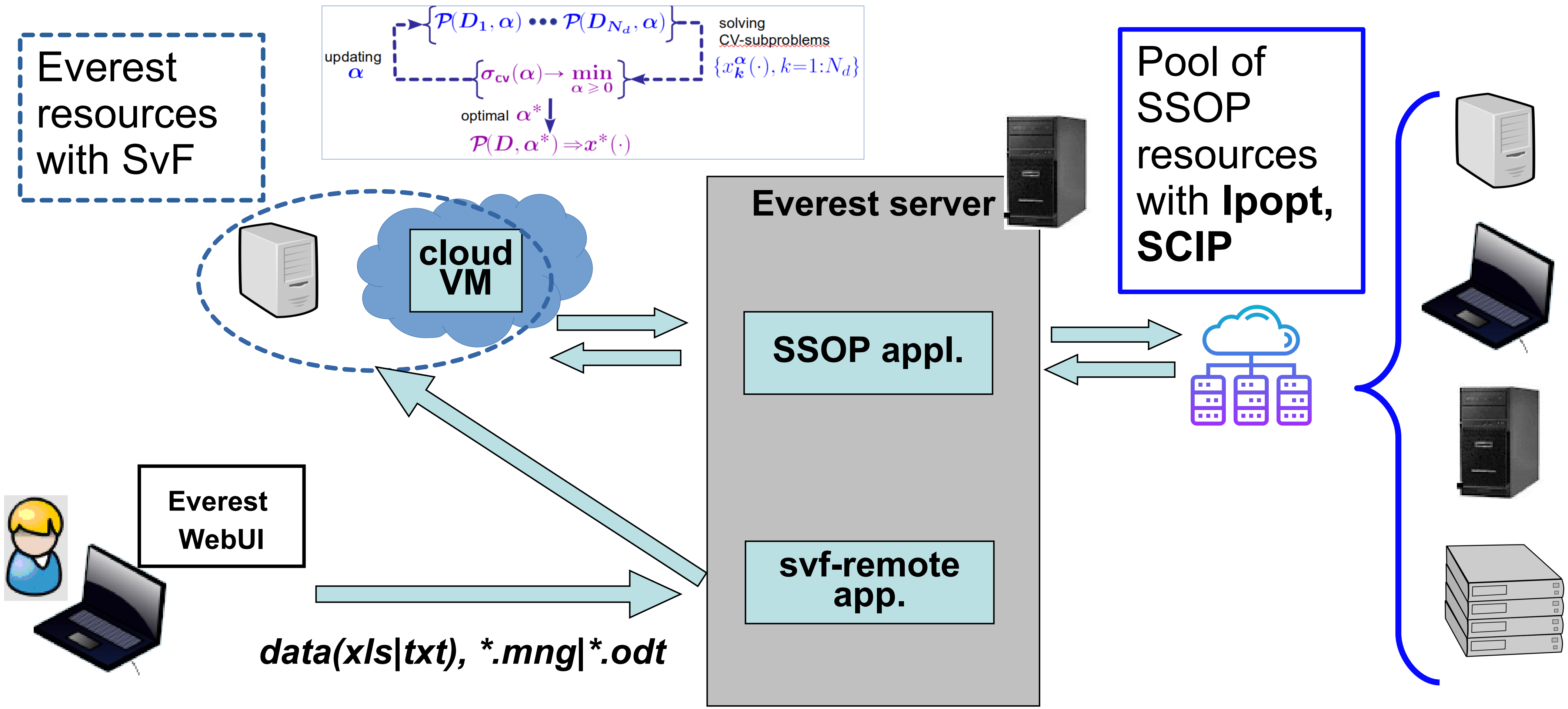
Everest user can select resource where SvF-remote job will be run

Requirements:

SvF-toolkit is installed and runSvF-remote.sh is in the $PATH



svf-remote service enables to use only browser to send data files and MNG-file with SvF-task to "executor". User can select resource to run ***svf-remote*** job.



Example of MNG-file, *.ODT format. Dumped oscillator.

```
Here you can place arbitrary text and pictures
BoF-SvF

Runmode = 'P&S'
SvF.DrawMode = 'File'                  #'File&Screen'
SvF.Resources = ["pool-scip-ipopt"] #["abc_pc" ]
CVNumOfIter     10 # Number of iter.
CVstep          21 # Numberof CV subproblems

Select x, t  from  Spring5.dat
GRID:
        t  ∈ [ -1.,  2.5, 0.025 ]

Var:        x ( t )
            v ( t )
            K
            μ     #  will be replaced to muu in Python code
            xr

SchemeD1 = Central
# EQ:    d2/dt2(x) == - K * ( x - xr ) - muu * v
#        v == d/dt(x)
EQ:
```

$$\frac{d^2}{dt^2}(x) == -K*(x-xr)-\mu*v$$

$$v == \frac{d}{dt}(x)$$

```
OBJ:    x.Complexity ( Penal[0]) + x.MSD()

Draw
EOF
Some remarks, figures, formulas may be added here
```

Conclusions and future plans

Conclusions

- ***svf-remote*** application reproduces all conveniences of current SvF-tookit:
  - symbolic formulation of identification task in MNG-file (ASCII text, OpenOffice *.odt) with symbolic formulas
  - access to all Everest resources with optimization solvers
  - returns results in numerical and graphical form
- Ease of use from browser for a long calculations on the principle of “launch and wait for notification by mail”
- You can run ***svf-remote*** on any Everest resources (servers, desktops, VMs) where SvF-toolkit is installed

Nearest future plans

- Involve our IPPI servers as **svf-remote** deployment hosts
- Migrate from Everest “local” Agent on Yandex Cloud VM to Everest YCC Agent (for Yandex Cloud Container) (to save money)
- To try Jupyter as another user interface to SvF-toolkit, installed on remote host (SSOP will remain with us)

Thank you.

Questions?

http://bit.ly/VoloshinovGScholar

vladimir.voloshinov@gmail.com

Discretization of autonomous DE: polynomila + mesh

One of approach in SvF: outer function is replaced with polynomial with unknown coefficients, inner function — unknown values on a meshgrid (by t):

$$F(x)\sim\mathcal{P}(x)=\sum_{p=0}^{P} c_p\cdot x^p \qquad x(t)\sim\{(x_i,t_k): i=0{:}N_x, k=0{:}N_t\},$$

$$x_i=x_{\mathbf{L}}+i\cdot\Delta x,\ \Delta x=\frac{x_{N_x}-x_{\mathbf{L}}}{N_x},$$

$$t_k=t_{\mathbf{L}}+k\cdot\Delta t,\ \Delta t=\frac{t_{N_t}-t_{\mathbf{L}}}{N_t},$$

$$\left\{\dot{x}=F(x(t))\right\}\sim\left\{\frac{x(t_k)-x(t_{k-1})}{\Delta t}=\sum_{p=0}^{P} c_p\cdot\left(\frac{x(t_k)+x(t_{k-1})}{2}\right)^p, k=1{:}N_t\right\}$$

$$\left\{\ddot{x}=F(x(t))\right\}\sim\left\{\frac{x(t_{k+1})-2x(t_k)+x(t_{k-1})}{\Delta t^2}=\sum_{p=0}^{P} c_p\cdot x(t_k)^p, k=1{:}(N_t-1)\right\}$$

$$x(t_d)\sim\widehat{x}_d=\frac{t_k-t_d}{\Delta t}x(t_{k-1})+\frac{t_{k-1}-t_d}{\Delta t}x(t_k),\ \ t_d\in[t_{k-1},t_k]$$

$$\frac{1}{D}\sum_{d=1:D}(\tilde{x}_d-\widehat{x}_d))^2+\alpha\sum_{\dots}\left(\frac{\mathcal{P}(x_{i+1})-2\mathcal{P}(x_i)+\mathcal{P}(x_{i-1})}{\Delta x^2}\right)^2\Delta x\rightarrow\min_{x_i,c_p}.$$

This we have **nonlinear mathematical programming problem** with polynomials of non-small degrees (up to 7, 8)

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