SAMARA UNIVERSITY

Implementing computations with dynamic task dependencies in the desktop grid environment using Everest and Templet Web

I. Bobyleva[1)], S. Vostokin[1)], S. Popov[1)], O. Sukhoroslov[2)]

[(1)] Samara University, [(2)] IITP RAS

Presenter: S. Vostokin,
Prof., Information Systems and Technologies Dept.,
Samara National Research University
(Samara University)

GRID 2018, Dubna, September 10-14

Provide a proof of technology solution (PoT) for desktop grid applications with complex (dynamic) dependencies between tasks.

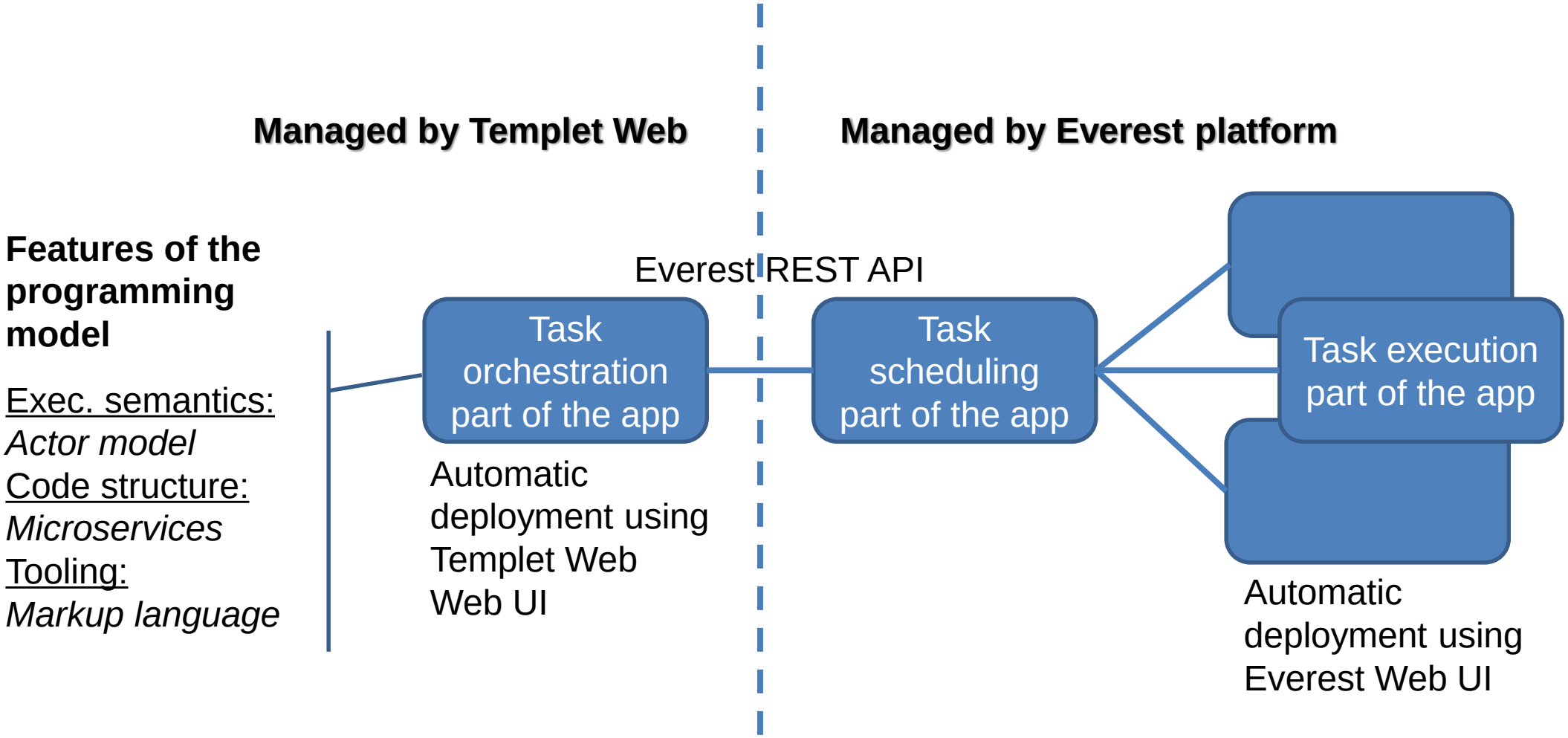
This kind of apps is in the demand in growing fields of science like Data Science, Neuroinformatics, Bioinformatics, and mathematical modeling of complex systems in general.

Features of	Desktop grid	Dedicated cluster
Low cost and availability	+	-
Easy to scale	+	-
Good for embarrassingly parallel problems	+	+
Good for problems with task dependencies	???	+

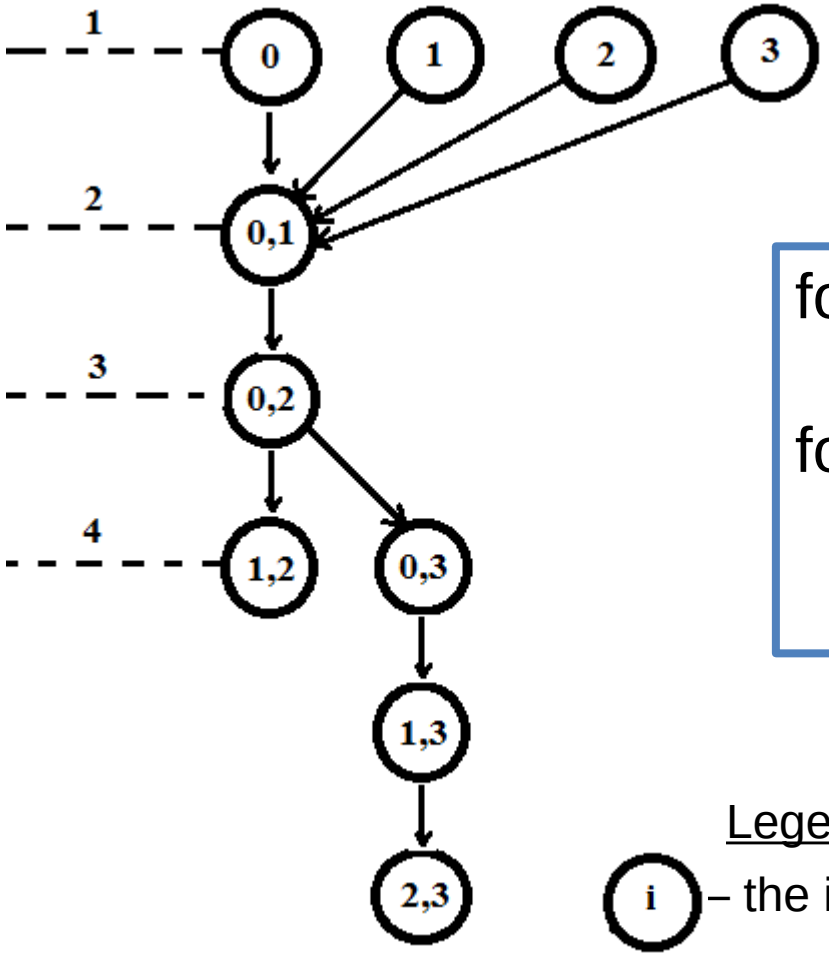
I. Develop a prototype application that uses dynamic task dependency graph (*based on the block sorting algorithm*).

II. Develop a scheme for deploying the application in the desktop grid environment (*running the Everest and Templet Web platforms*).

III. Do the experimental study of the possibility of fault-free calculations with a large number of interdependent tasks (*for the proposed application architecture and deployment scheme*).


Managed by Templet Web
Managed by Everest platform
Features of the programming model
Exec. semantics:
Actor model
Code structure:
Microservices
Tooling:
Markup language
Everest REST API
Task orchestration part of the app
Task scheduling part of the app
Task execution part of the app
Automatic deployment using Templet Web Web UI
Automatic deployment using Everest Web UI

- The app should make the use of idle computers
- The app should execute in a heterogeneous environment
- The app should have a simple and rapid deployment
- The app should provide long term fault-tolerant computing
- The app should manage a large number of interdependent tasks




```
for (int i = 0; i < N; i++)
    block_sort(i);
for (int i = 1; i < N; i++)
    for (int j = 0; j < l; j++)
        block_merge(j, i);
```

Legend for the task dependencies graph:

i – the i-th block sorting;

i,j – the i-th and the j-th blocks merging

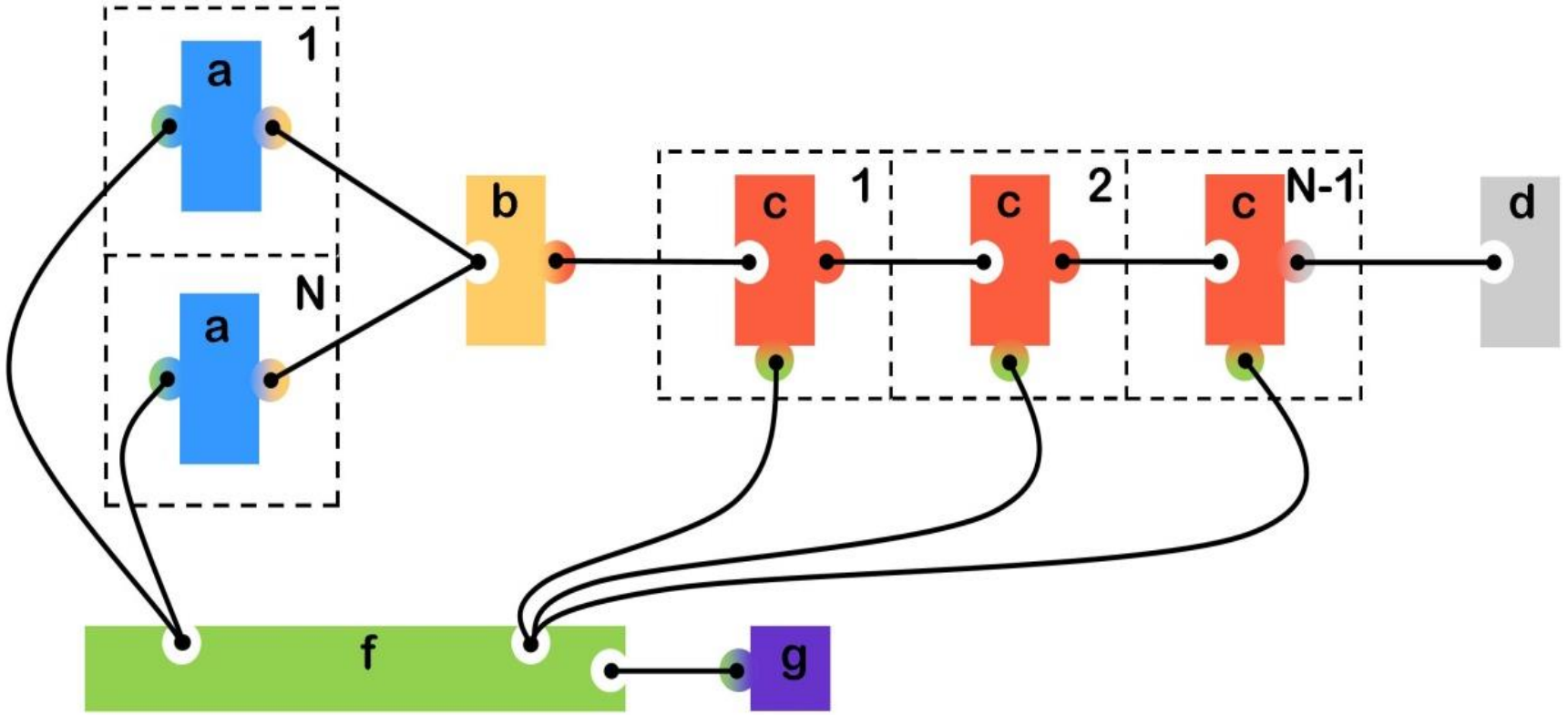
Why we choose this problem?

The problem statement is simple for the sequential computation.

The problem simulates operations relevant to many application domains:

- finding a correlation between data items;
- finding a frequency of data items;
- ordering of data items.

It is easy to vary the problem complexity and the number of tasks in experimental study.



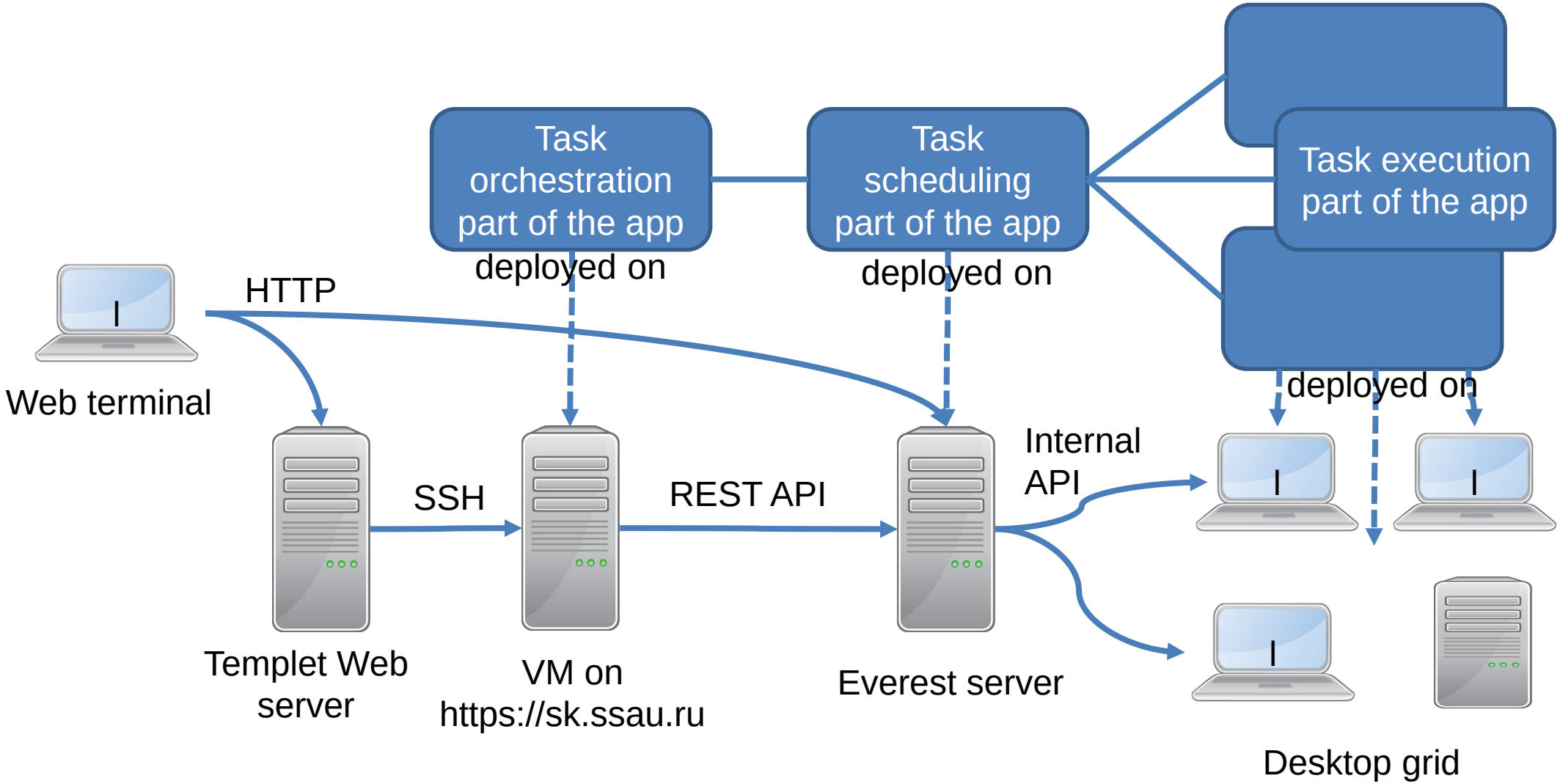
Microservices:

a – sorting;

c – merging;

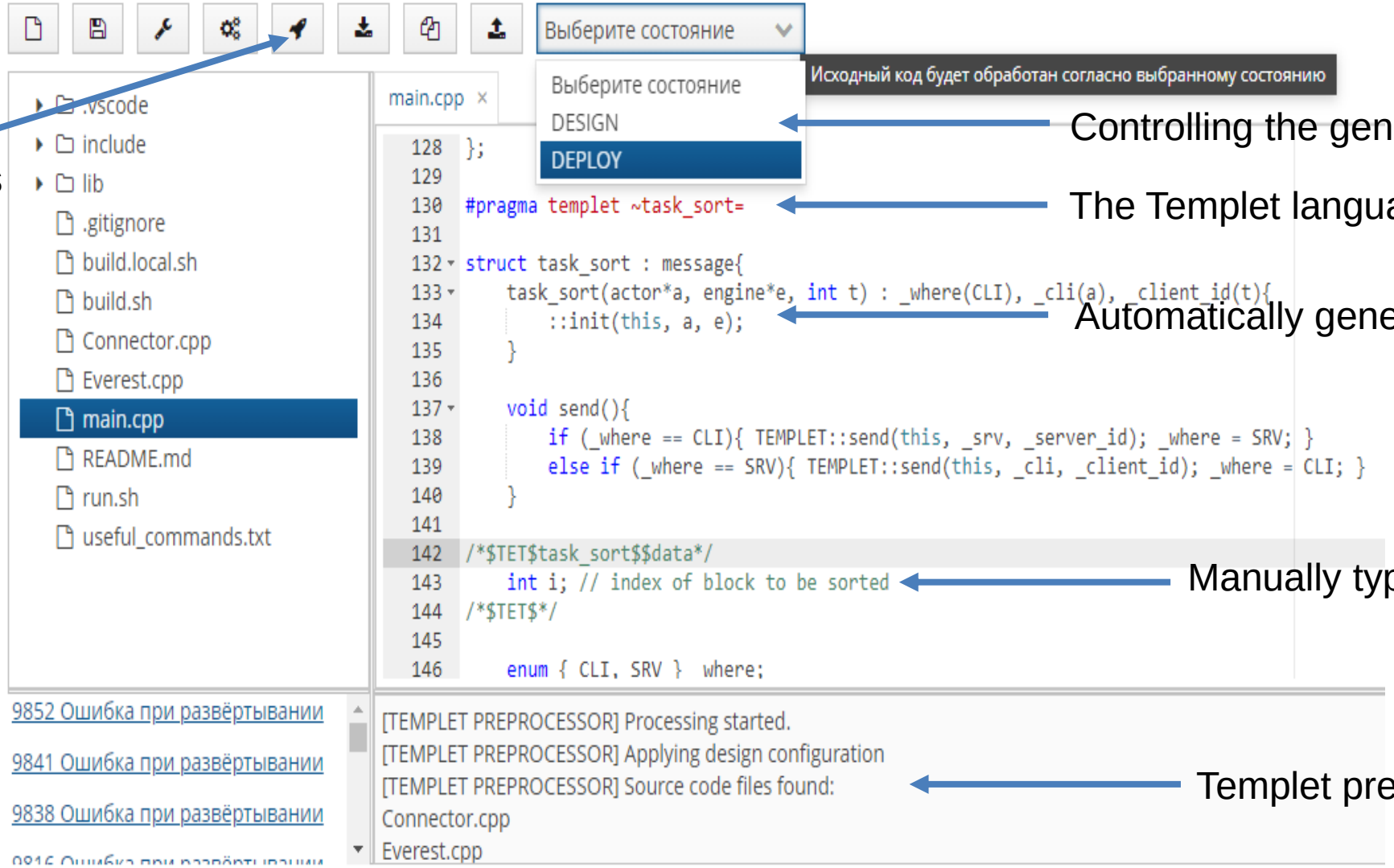
f – interaction with Everest platform;

b, d, g – ancillary.


Task orchestration part of the app
Task scheduling part of the app
Task execution part of the app
deployed on
deployed on
deployed on
HTTP
Web terminal
SSH
REST API
Internal API
Templet Web server
VM on https://sk.ssau.ru
Everest server
Desktop grid

PROGRAMMING THE APP IN THE TEMPLET WEB IDE

By pressing this button you deploy and run the app



Controlling the generation of code

The Templet language markup

Automatically generated code

Manually typed code

Templet preprocessor output

sorter

About | Parameters | Submit Job | Presets | Discussion

Inputs

	Title	Name	Type	Values	Default	Description
✔	file	file	URI			Input file

Outputs

	Title	Name	Type	Description
✔	StdErr	stderr	URI	
✔	outSort	outSort	URI	Output file

merger

About | Parameters | Submit Job | Presets | Discussion

Inputs

	Title	Name	Type	Values	Default	Description
✔	file1	file1	URI			First block
✔	file2	file2	URI			Second block
✔	i	i	string			i index
✔	j	j	string			j index

Outputs

	Title	Name	Type	Description
✔	StdErr	stderr	URI	
✔	outMerge1	outMerge1	URI	Output file
✔	outMerge2	outMerge2	URI	Output file

Jobs

Auto Update | Update | Filters

Name	Application	State	Owner	Submitted	Finished	Data Size	Actions
sorter	sorter	RUNNING	stefanpopov	09 Jun 2018 11:08:45			■
sorter	sorter	DONE	stefanpopov	08 Jun 2018 16:39:21	08 Jun 2018 16:39:37	1.96 KB	🗑
sorter	sorter	DONE	stefanpopov	08 Jun 2018 16:39:03	08 Jun 2018 16:39:21	1.96 KB	🗑
sorter	sorter	DONE	stefanpopov	08 Jun 2018 16:38:46	08 Jun 2018 16:39:03	1.96 KB	🗑
sorter	sorter	DONE	stefanpopov	08 Jun 2018 16:38:30	08 Jun 2018 16:38:46	1.96 KB	🗑

```
NUM_BLOCKS = 8
BLOCK_SIZE = 160000
OMP_NUM_PROCS = 2
Token: oa5340ipqiokpats7r221e1543zci85961zmkh5om37az4zvxm8eu0a3ec3u00y1

Block-sort time is 315.078 sec

is sorted = true

Array blocks available at following links:
https://everest.distcomp.org/api/files/jobs/5b20b618100000e124e12a36/0/outMerge1-0
https://everest.distcomp.org/api/files/jobs/5b20b6351000009012e12a45/0/outMerge1-1
https://everest.distcomp.org/api/files/jobs/5b20b64e1000009012e12a4f/0/outMerge1-2
https://everest.distcomp.org/api/files/jobs/5b20b6601000005e25e12a59/0/outMerge1-3
https://everest.distcomp.org/api/files/jobs/5b20b671100000561ce12a63/0/outMerge1-4
https://everest.distcomp.org/api/files/jobs/5b20b6791000005b11e12a68/0/outMerge1-5
https://everest.distcomp.org/api/files/jobs/5b20b6811000005b11e12a6d/0/outMerge1-6
https://everest.distcomp.org/api/files/jobs/5b20b6811000005b11e12a6d/0/outMerge2-7
```

Number of blocks	Number of merge tasks	Number of sort tasks	Total tasks
2	1	2	3
4	6	4	10
8	28	8	36
16	120	16	136
32	496	32	528
64	2016	64	2080
128	8128	128	8256

Number of blocks	Block size, KB	Execution time, s
2	640	31,18
4	320	83,44
8	160	315,08
16	80	1019,85
32	40	3755,36
64	20	14580,80
128	10	54076,76

SUMMARY OF METHODS USED TO ACHIEVE THE APPLICATION REQUIREMENTS

- **the use of idle computers** → environment for task execution included notebooks, desktop computers, and virtual machines
- **execution in a heterogeneous environment** → we used Linux for orchestration part and Windows (Linux is also possible) for task execution part of the application
- **simple and rapid deployment** → was provided by the Everest and Templet Web UI
- **long term fault-tolerant computing** → the orchestration part used standalone virtual machine and the task execution part was controlled by the Everest
- **a large number of interdependent tasks** → actor-based programming model adapted for task management

A PoT solution for desktop grid applications with complex dependencies between tasks was developed.

The perspective of using the actor-based programming model for task orchestration together with advanced task management platform was proved.

We plan to extend the Templet framework code for transparent interaction with the Everest platform, making it easy to write applications.

SAMARA UNIVERSITY

THANK YOU

Feel free to contact:
Sergey.Vostokin@gmail.com