SAMARA UNIVERSITY

Sergei Vostokin
Samara National Research University

USING THE ACTOR MODEL FOR FLEXIBLE TASK MANAGEMENT IN DISTRIBUTED COMPUTING ENVIRONMENTS:

A REVIEW OF THE TEMPLET PROJECT AND THE POSSIBILITY OF ITS INTEGRATION INTO THE SHARED-USE ENVIRONMENT OF THE NICA COMPLEX.

VI SPD Collaboration Meeting and Workshop on Information Technology in Natural Sciences, Samara University, 23–27 Oct 2023

OUTLINE

1. **About the Templet project.**
2. **The Software Development Kit sub-project features.**
3. **Use cases of apps for non-dedicated computing environment.**
4. **Proposals for technical cooperation.**

ABOUT THE TEMPLET PROJECT

- ✓ Motivation - we learn to use Internet computing power.
- ✓ Main approach. About the name of the project.
- ✓ History of development and web resources of the project.

Motivation - we learn to use Internet computing power

As the volume of computing (*AI*, *big data*, *computer simulation*) **grows**, so does **the need for programs that can effectively use idle computing resources** within the enterprise or on the Internet.

This kind of programs is practically important because:

- ❑ they can get a **large amount of computing resources**,
- ❑ a **lower cost** than using dedicated systems.

This kind of programs is tricky and fun to develop because:

- ❑ **fault tolerance**,
- ❑ **load balancing** and other issues of distributed systems to be solved.

Motivation - we learn to use Internet computing power

Where is the Internet's idle computing power?

- **Volunteer computers**, as in the *BOINC* project or other voluntary distributed computing projects.
- Temporarily **idle corporate computers** that are potentially available over the network to solve production problems, as in the *HTCondor* project.
- Temporarily **free computing nodes** of high performance supercomputer or cluster systems.
- Free or low cost **virtual machines** *(spot VMs*) or **serverless runtimes** from cloud providers.

Main approach. About the name of the project.

Unlike projects (*BOINC*, *HTCondor*) that focus on developing infrastructure or middleware, **we focus on how to build programs on top of existing grid and/or cloud infrastructure**.

We follow the approach of **isolating the computation engine from the application logic**, called **algorithmic skeletons**, proposed by *Murray Cole*.

This skeleton or template or **Templet** is a **special form of code structure** that we use to program on idle resources.

History of development and web resources of the Templet project

- Project **Wiki** (templet.ssau.ru/wiki)
  - since August, 2013 (active)
- Registered **trademark** (Rospatent)
  - since January, 2014 (active)
- **TempletWeb** service (templet.ssau.ru/app) –
  - since June, 2015 (retired in 2022)
- **TempletSDKx2** (https://github.com/templet-language)
  - since August, 2016 (retired)
- **TempletSDKx3** (https://github.com/the-templet-project)
  - since August, 2020 (active)

TEMPLET

It has been successfully used to **teach courses** and **conduct research** in distributed and parallel computing (with **TempletWeb** >1000 students worked on idle nodes of SK cluster).

THE SOFTWARE DEVELOPMENT KIT SUB-PROJECT FEATURES

- ✓ Using actor model to submit jobs.
- ✓ New kind of actor model with global state.
- ✓ Using a metalanguage to simplify programming.
- ✓ One application - many runtime libraries.

Using actor model to submit jobs

The basic principle of computing in a faulty environment is well known – **dividing computation into small portions** (jobs or tasks).

- ✓ *This is also typical for processing experimental data in various fields, including nuclear physics.*

Usual approaches for this include:

- ❑ creating all tasks beforehand;
- ❑ using DAG-based languages for exposing task dependances;
- ❑ using frameworks such as BOINC, etc.

We adapt **the actor model** proposed by *Carl Hewitt* to perform tasks at “isolated steps” of actor execution. This is **the key for flexible task management**.

New kind of actor model with global state

Actor model has a source in functional programming. But we believe that

the algorithmic representation of actors is more convenient for programmers.

Our variant of actors:

- ❑ need no special syntax to isolate actor state;
- ❑ can be considered as an algorithm with a predefined structure;
- ❑ can be thought of as *Cole's algorithmic skeleton*.

The **book** shows how to use the programming model for coding applications.

Using a metalanguage to simplify programming

Maintaining the correct code structure for an actor model is hard even when using the popular Akka actor implementation.

To solve this problem, **we use a metalanguage** to describe the structure of the code.

The metalanguage processor:

- ❑ automatically generates the correct code structure;
- ❑ seamlessly integrates the runtime with application code.

The **demonstration** shows how to program using the metalanguage processor.

One application - many runtime libraries

To date, four runtime libraries have been implemented:

- ❑ sequential execution of tasks for logical debugging (**base**);
- ❑ simulating a time delay to debug the performance (**basesim**);
- ❑ parallel execution of tasks in a shared memory (**omptask**);
- ❑ distributed execution of tasks over the Internet (**everest**).

✓ *Everest platform, developed by IITP RAS, is used as grid middleware.*

Running the same code in different environments allows:

- ❑ to debug a distributed application the same way as a sequential application;
- ❑ to get an earlier idea of possible speedup without running on multiple nodes;
- ❑ to use whatever middleware you prefer with job execution semantics.

USE CASES OF APPS FOR NON-DEDICATED COMPUTING ENVIRONMENT

- ✓ The computationally intensive application:
  analysis of dynamical system by calculating the Lyapunov exponent
- ✓ The data-intensive application:
  frequency analysis of microblogging

Analysis of dynamical system by calculating the Lyapunov exponent

The computationally intensive application features.

- Demonstration of interoperability and sharing licenses: calculations of Lyapunov exponents were carried out in the Maple language.
- There was a group of computers and virtual machines with Maple licensed packages installed.
- We combined these computers to calculate Lyapunov exponents in parallel and automated the issuance of tasks for them.

Source: *Popov S.N. , Vostokin S.V., Doroshin A.V. Dynamical systems analysis using manytask interactive cloud computing // Journal of Physics: Conference Series. 2020. Vol. 1694. Issue 1.*

The data intensive application features.

- Demonstration of the ability to process data in non-dedicated computing environments.
- Demonstration of programming algorithms with complex dependencies between tasks.
- To store and exchange data between virtual machines in the public cloud of Samara University, access to a distributed file system was used.

Source: *Vostokin S., Bobyleva I. V. Implementation of frequency analysis of Twitter microblogging in a hybrid cloud based on the Binder, Everest platform and the Samara University virtual desktop service // CEUR Workshop Proceedings. 2020. Vol. 2667. P. 162-165.*

PROPOSALS FOR TECHNICAL COOPERATION

- ✓ Deployment on the JINR MLIT testbed resources.
- ✓ Adding DIRAC job management to the SDK.
- ✓ Development of a runtime/middleware in a pilot-job paradigm.

Deployment on the JINR MLIT testbed resources

Current deployment

- ❑ TempletSDKx3 source code → GitHub.com
- ❑ TempletSDKx3 in JupyterLab → MyBinder.org
- ❑ Distributed execution control → Everest.distcomp.org

Deployment for testing at JINR

- ❑ TempletSDKx3 source code → HybriLIT/GitLab
- ❑ TempletSDKx3 in JupyterLab → HybriLIT/BinderHub
- ❑ Distributed execution control → Everest.distcomp.org or DIRAC

Adding DIRAC job management to the SDK

Current job management

- Job submission - REST, HTTPS/JSON using libcurl (C/C++)
- Job execution - Everest agent (Python)
- File transfer - Everest platform, using libcurl (C/C++), Windows Server

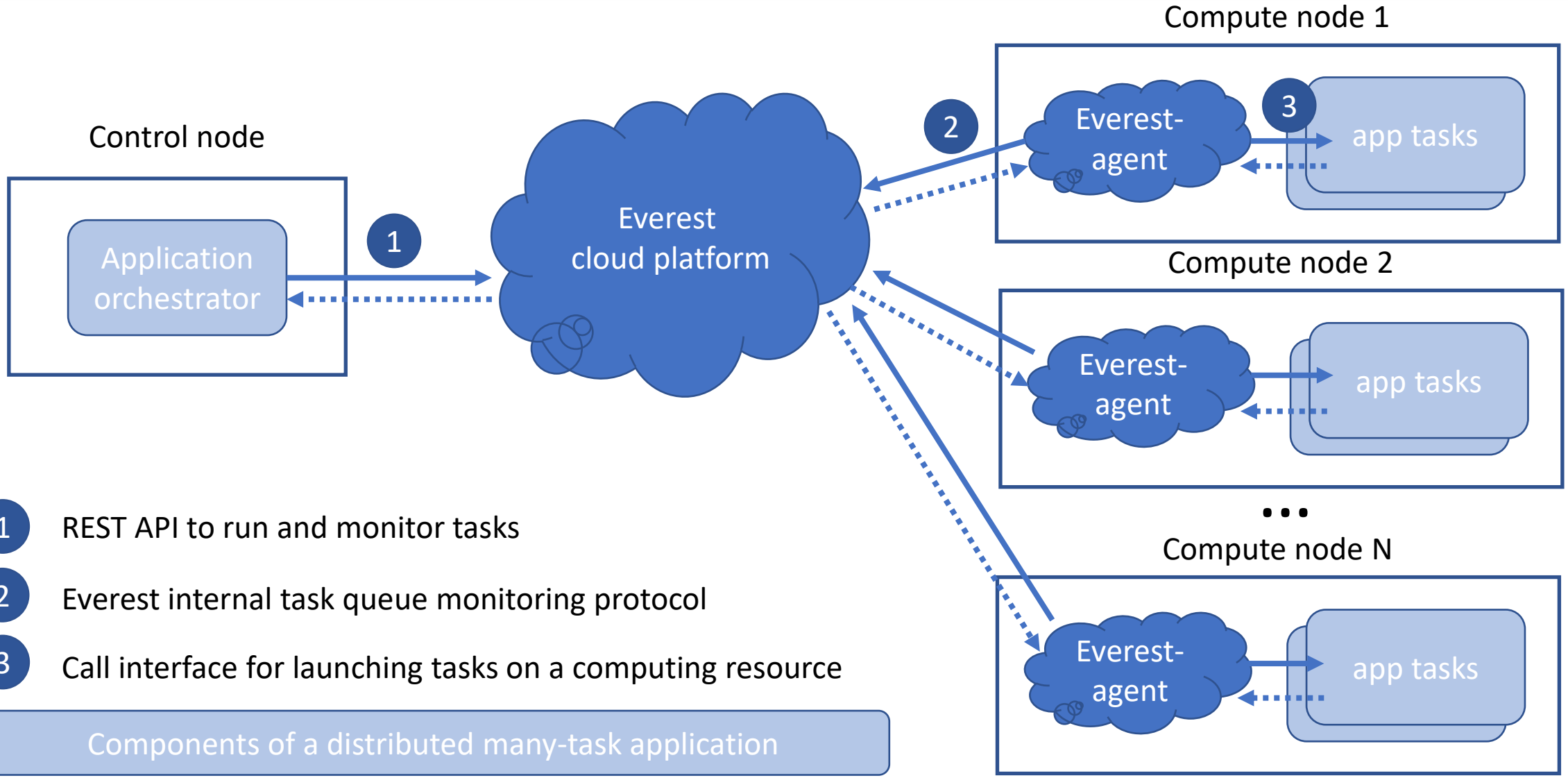
Everest

Job management for testing at JINR

- DIRAC via REST using libcurl (C/C++)
- required polling time from 1 to 5 seconds

DIRAC
THE INTERWARE

Development of a runtime in a pilot-job paradigm (current architecture)



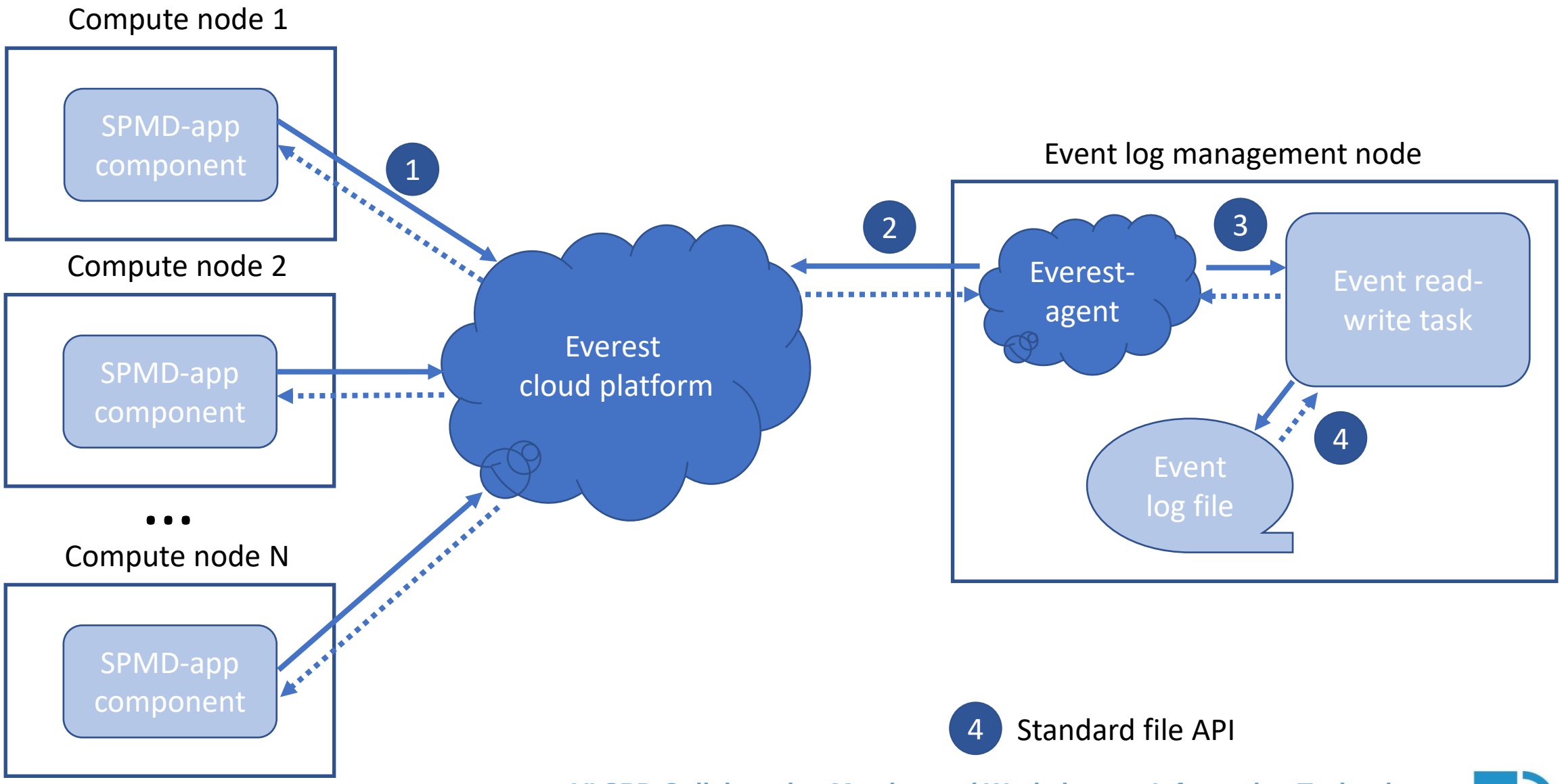
1 REST API to run and monitor tasks

2 Everest internal task queue monitoring protocol

3 Call interface for launching tasks on a computing resource

Components of a distributed many-task application

Development of a runtime in a pilot-job paradigm (proposed architecture)



THANK YOU FOR YOUR ATTENTION !

http://templet.ssau.ru - главная страница

http://templet.ssau.ru/wiki - вики проекта Templet и образовательные ресурсы

https://github.com/the-templet-project - Templet SDK x3 - актуальная версия

Автор: Востокин Сергей Владимирович

д.т.н., зав. кафедрой программных систем, Самарский университет

easts@mail.ru

СПАСИБО ЗА ВНИМАНИЕ !