EXPLORING APPLICATIONS AND OPPORTUNITIES OF REMOTE VIRTUAL SUPERCOMPUTER

O. Iakushkin[1,2,a], D. Malevanniy[1], O. Sedova[1], A. Degtyarev[1,2], V. Korkhov [1,2]

[1] *St Petersburg University*

[2] *Plekhanov Russian University of Economics Stremyanny lane, 36, Moscow, 117997, Russia*

E-mail: [a] o.yakushkin@spbu.ru

This paper focuses on integrating a set of technologies such as Blockchain development platforms and HPC applications such as Schrödinger smoke rendering into remote virtual supercomputer system. We have successfully run GPGPU and CPU heavy workloads, User interface heavy and Terminal based applications. We have successfully created a set of virtual work environments for: Telegram Open Network, Hyperledger Fabric, Ethereum, Corda and Iroha blockchains; Student summer AI competitions, Server-side rendering of Schrödinger smoke.

Keywords: Blockchain, Docker, Virtual Supercomputer, Remote Desktop, Education

Oleg Iakushkin, Daniil Malevanniy, Olga Sedova, Alexander Degtyarev, Vladimir Korkhov

1. Introduction

We have developed a virtual desktop infrastructure (VDI) framework that allows users to gain simultaneous access to GPGPU and high-performance computational resources [1]. The main idea is to provide each user with a fully self-contained desktop environment including all SDKs, IDEs and applications needed. Our tool is best suited for long lived development processes that require a lot of short-lived computations running on rare shared resources such as GPGPU nodes. It simplifies and radically shortens time required for a user to get to the main activity such as software use or development by providing a familiar full-fledged desktop interface.

In other words, the main goal of the tool we created is to deliver a fast and reproducible approach for creating and sharing hand-tailored development environments. This allows a fast classroom and work infrastructure deployment, access and sharing, which in turn facilitates knowledge transfer.

Let's look at an example case: a junior researcher needs to access GPGPU development environment. If we point him to a classical cluster-based setting, he would need to learn in addition to his main development target how to work his way around terminal multitasking, cluster queues and cluster architecture basics, task management systems such as *slurm*, probably cluster package management system such as Singularity, how to debug programs running in a cluster. This is a long and tedious learning curve that will take at least two month and heavily impact time that could be spent on valuable research tasks.

A user in that case had to fully embrace HPC way first and only than could he approach his main software development task. We want to radically simplify the way a developer prototypes, tests and debugs his HPC and service applications: provide a remote accessible desktop interface with short-time access to mission critical resources such as GPGPUs. Such environments can provide the same level of user isolation as one can get in a Singularity container combining it with an ability to visually debug programs and use GUI based tools directly on a cluster node. Such approach removes stiff HPC learning curve that is usually presented to the developers of scientific applications and services, allowing for a more gradual acquaintance with HPC environment.

We created a VDI that provides users with access to computational environments nested deep inside the corporate infrastructure; with independent from computation nodes storage nodes; and a controller node that manages access rights and provides service access. Architecture is presented in the figure 1.

2. Blockchain development environments

We successfully integrated a set of different Blockchain environments into our system. Each environment is presented in form of a single runnable container image. Combining a base image with all the different required tools (listed below) and testing suites (see figure 2) preinstalled allows us users not only to develop and test new services but also fine-tune and compare capabilities of deferent distributed ledgers.

Corda. Inside this container, we have fitted all necessary instruments needed to start developing Corda based smart-contracts. There are:

1. Preconfigured development environment - IntelliJ IDEA
2. Prepared for Corda development OpenJDK8 and OpenJFX8
3. Our product: UI Tool. This tool is made to make demonstrations of command-line interface programs
4. A documented example of Corda-based smart-contract
   An example of a demo application with 5 nodes can be seen on Fig. 1.

Telegram Open Network (TON). Inside this container, we have fitted all necessary instruments needed to start developing smart-contracts for Telegram Open Network (TON) blockchain. There are:

1. Prebuilt tools and compilers for fift, funC languages.
2. Prebuilt and ready for production blockchain client.
3. Visual Studio Code Editor

Hyperledger/Fabric. Inside this container, we have fitted all necessary instruments needed to start developing HL Fabric based smart-contracts. There are:

1. Visual Studio Code editor with Hyperledger Composer plugin for developing your smart-contracts.
2. Testing network with all necessary dependencies (Docker, Go)
3. Our product: UI Tool. This tool is made to make demonstrations of command-line interface programs

For Hyperledger Fabric setup it is important to have local capabilities to run docker images, shown on Fig. 2.

Ethereum. Inside this container, we have fitted all necessary instruments needed to start developing Ethereum based smart-contracts. There are:

1. Geth console that runs the test network. Launches on container start, with mining turned on and one registred user with some ether.
2. Remix IDE - development environment for Ethereum smart-contracts, which can connect to the test network. Inside you will find a Solidity language editor and compilator, UI for interaction with smart-contracts, and a JavaScript console to interact with test network.
3. Standalone Solidity compilator - solc.
4. Smart-contract demo as an example of developing custom smart-contract and interaction with it.

3. Environments for research and education

For the development of Schrödinger smoke rendering application, we have equipped a node with a Mesa renderer and Houdini 3D Procedural Animation Software on top of it with OpenGL 3.3 emulation starting rendering scripts with:

```
MESA_GL_VERSION_OVERRIDE=3.3 scrypt.py
```

Such environment creation tool has quickly found use in a classroom allowing us to create and deploy student's virtual desktop environments in a protected setting, providing them with direct access to GPGPU nodes and powerful HPC nodes. Students have used this tool at:

1. SPbU summer coding school at 2018 and 2019 years
2. Sochi Sirius blockchain intensive course 2019

Examples showing developer IDE and 3D rendering application can be found in Fig. 3

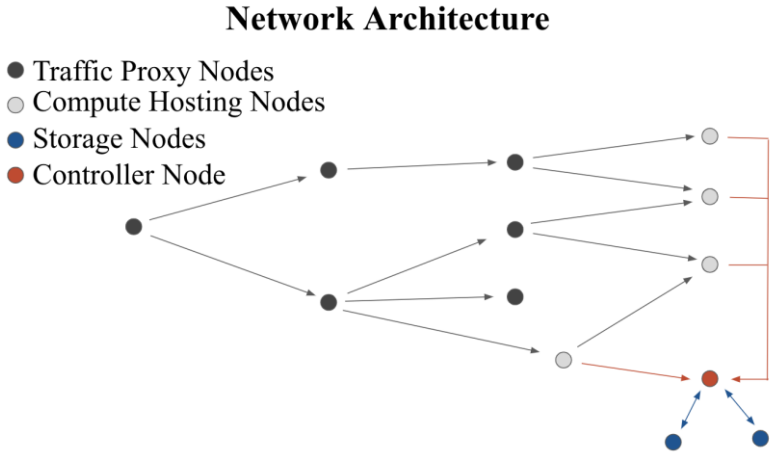

Figure 1. A VDI architecture of our solution

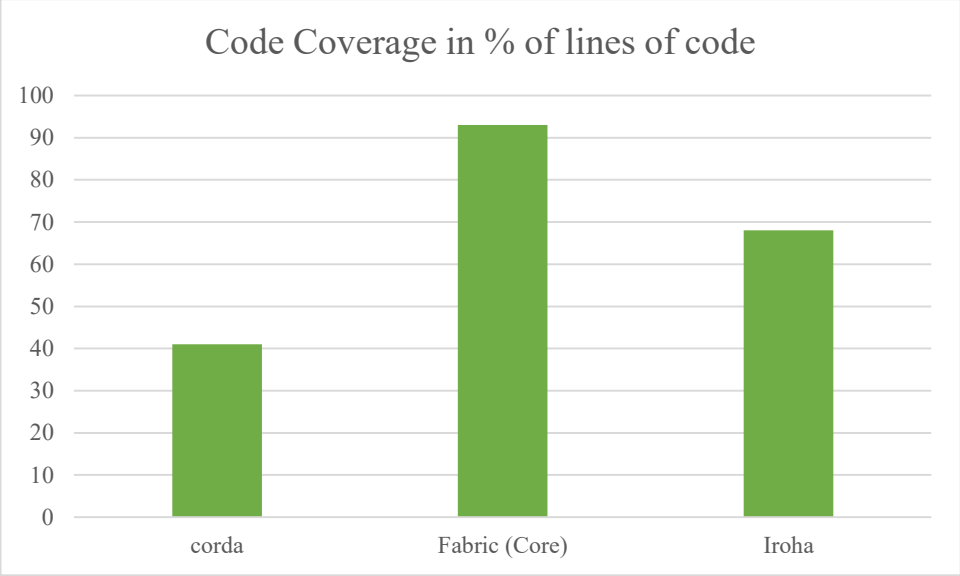

Figure 2. Comparison of Code Coverage results obtained via our container images

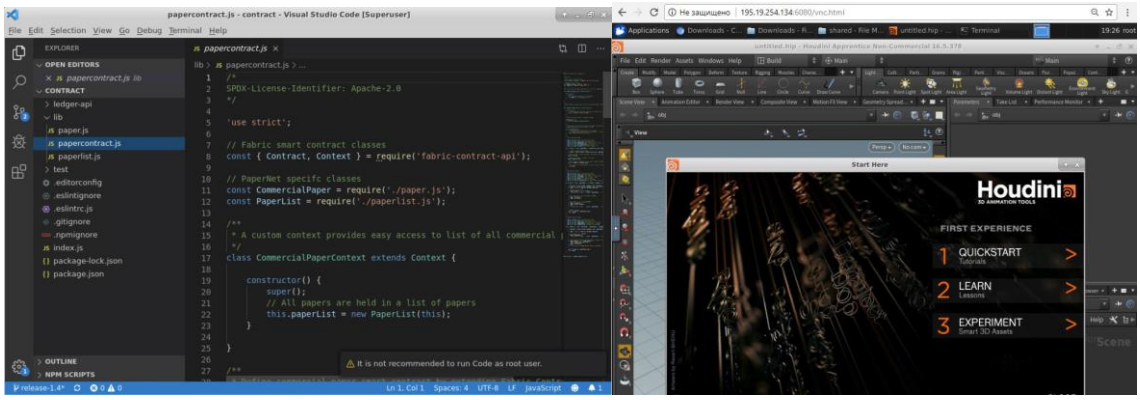

Figure 3. An IDE with a smart contract and a Houdini rendering application running in our VDI environment and remotely accessed via a web browser

3. Drawbacks

The main system drawbacks are space consumption per image and inability to equally share rare resources such as GPGPUs. This leads to the enforcement of computations running on such resources to be short lived. In future we hope to integrate better GPGPU provisioning such as Nvidia GRID to share GPGPUs.

4. Conclusion

The system we presented allows for multidisciplinary application deployment and shows interesting results for students training. We hope to see integration with applications [2-5] in the nearest future.

5. Acknowledgement

This research was partially supported by the Russian Foundation for Basic Research grants (projects no. 17-29-04288). The authors would like to acknowledge the Reviewers for the valuable recommendations that helped in the improvement of this paper.

References

[1] Malevanniy D., Sedova O., Iakushkin O. (2019) Controlled Remote Usage of Private Shared Resources via Docker and NoVNC. In: Misra S. et al. (eds) Computational Science and Its Applications – ICCSA 2019. ICCSA 2019. Lecture Notes in Computer Science, vol 11622. Springer, pp. 782-791

[2] Iakushkin O., Selivanov D., Tazieva L., Fatkina A., Grishkin V., Uteshev A. (2018) 3D Reconstruction of Landscape Models and Archaeological Objects Based on Photo and Video Materials. In: Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA 2018. ICCSA 2018. Lecture Notes in Computer Science, vol 10963. Springer, pp. pp 160-169

[3] Iakushkin O., Fatkina A., Plaksin V., Sedova O., Degtyarev A., Uteshev A. (2018) Reconstruction of Stone Walls in Form of Polygonal Meshes from Archaeological Studies. In: Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA 2018. ICCSA 2018. Lecture Notes in Computer Science, vol 10963. Springer, pp 136-148

[4] Iakushkin, Oleg, Anna Kondratiuk, Alexey S. Eremin, and Olga Sedova. "Development of a containerized system to build geometric models and perform their strength analysis." In 3rd International Conference on Applications in Information Technology, ICAIT 2018, pp. 146-149. Association for Computing Machinery, 2018.

[5] Iakushkin, Oleg O., and Olga S. Sedova. "Creating CAD designs and performing their subsequent analysis using opensource solutions in Python." In AIP Conference Proceedings, vol. 1922, no. 1, p. 140011. AIP Publishing, 2018.