Traffic Flow Optimization in Smart Cities
Leveraging HPC Resources for Automatic Design Space Exploration

Martin Šurkovský, Jakub Beránek
Kateřina Slaninová, Jan Martinovič
IT4Innovations
VSB - Technical University of Ostrava
17. listopadu 2172/15
Ostrava, Czech Republic
Email: martin.surkovsky@vsb.cz, jakub.beranek@vsb.cz
katerina.slaninova@vsb.cz, jan.martinovic@vsb.cz

Radim Cmar
Sygic a.s.
Twin City C, Mlynské Nivy 16
Bratislava, Slovakia
Email: rcmar@sygic.com

1. Introduction

In recent years, one of the major challenges for today's computing system is big data analytics. Therefore, it seems that future Big Data systems will be data-driven, featuring complex heterogeneous architectures that must be redesigned or customized based on the nature and locality of the data, and the type of learning/decisions to be performed. Programming environments able to concurrently co-design computation (with an efficient customization of the single node) and communication (with a proper optimization of the data movements among nodes) will effectively enable the use of heterogeneous, distributed, scalable and secure systems for Big Data analytics.

The EVEREST (dEsign enVironmEnt foR Extreme-Scale big data analyTics on heterogeneous platforms) project [1] aims at developing a holistic design environment that addresses simplifying the programmability of heterogeneous and distributed architectures for Big Data applications. The project uses "data-driven" design approach with domain-specific language extensions, hardware-accelerated AI and an efficient monitoring of the execution with a unified hardware/software paradigm.

One of the three applications selected for the validation of the EVEREST approach is a traffic modeling framework for intelligent transportation in smart cities. This framework combines a traffic simulator, a traffic prediction model and intelligent routing methods to better characterize the road traffic based on combination of multiple data in order to reduce congestions in the traffic infrastructures.

2. Defining and Executing Workflow Pipelines in Large-Scale Distributed Environments

HyperQueue [2] is a task execution library designed for transparent execution of tasks over HPC job managers like PBS/Torque or Slurm. It allows users to submit computational tasks in a simple way, without having to manually use the HPC job manager. HyperQueue is able to automatically submit HPC job allocations on behalf of the user, thus saving them from a lot of manual work. It is also designed to be efficient; its overhead per task is below 1ms.

One of the main benefits of HyperQueue in the context of the EVEREST project is its support for defining complex generic resources. It allows users to attach arbitrary resources to computational providers (workers) and also to request fine-grained resources for individual tasks. This feature can be leveraged for example in heterogeneous clusters, one of the designed use-cases for EVEREST, by assigning certain tasks to nodes containing FPGA (Field Programmable Gate Arrays) accelerators. For the traffic simulator use case it was used to execute a hyper-parameter search, which tested a series of routing parameters to find how they affect the simulation of cars driving around within a city.

There are several other task graph distributed execution tools [3]–[8]. HyperQueue is unique in its ability to schedule tasks with fine-grained generic resources and also thanks to its automatic allocator, which is able to automatically create HPC jobs on behalf of the user.

HyperQueue is a follow-up to HyperLoom [9], a platform for defining and executing workflow pipelines in large-scale distributed environments presented at SC 2017 in a research poster section [10], [11].

3. Deterministic Traffic Simulator

Deterministic traffic simulator is a new version developed within the EVEREST project. The traffic simulator is based on a smart navigation system that was developed in the EU funded project ANTAREX and recognized by European Commission as the innovation radar finalist for an innovation ready for market [12]. The decision to create a deterministic version was made particularly to have more control over the simulation and possibility to reproduce the

Figure 1. Snapshot of Traffic Flow in Prague, Czech Republic Jun 16 2021, 8:10 AM.

results. Thus we can fine tune specific parts of the simulator and its algorithms.

The traffic simulator simulates the movement of vehicles on a map over a time period. The simulator takes as an input an O/D (origin/destination) matrix which specifies a list of vehicles with origin and destination GPS, plus a departure time. With the traffic simulator we can investigate the traffic flow within a city and state questions such as *"What happens if this bridge is closed?"* or *"What if there is a traffic jam?"*. Vehicles move in regular time steps. For each step a new rouging is computed from the current position. Each vehicle regularly ask for re-routing, e.g. every 20s. One of the main goals is to optimize the entire traffic flow within a city. In other words, to distribute the flow in a way that the number of traffic jams is minimized. For this purpose we take advantage of *Probabilistic Time Dependent Routing* (PTDR) [13]. An example of a simulation snapshot can be seen in Fig. 1 where the red routes indicate traffic congestion. The thicker the line is the higher congestion on a route.

To setup the simulator, however, is not straightforward as the simulator takes a set of parameters that influence quality of the results and the computation time. This poster presents the usage of HyperQueue as an ensemble tool that manages HPC resources and performs various simulations with different settings. The results of simulations are discussed as well.

4. Simulator's parameters space exploration

The goal is to identify the most appropriate setting for the simulator that provides best results in sense of quality and performance. The performance is also influenced by the input data; longer routes prolong the time of the simulation.

The experiments are performed over Prague, Czech Republic at Jun 16 2021 between 7 AM to 10 AM. The results are investigated between 8 AM to 9 AM to cover morning rush hour. Time before and after investigated period can be seen as a warming and cooling phase of the simulation, respectively.

The initial state of simulation (O/D matrix) is based on anonymised information from mobile devices provided by T-Mobile operator.

To find the most appropriate setting for the simulator, a set of 75 variants was generated. Each variant specifies a

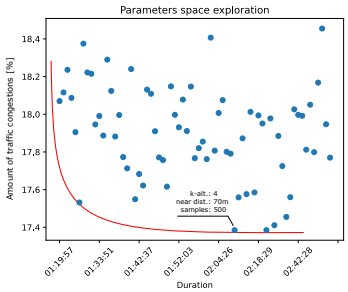

Figure 2. Results of parameter exploration.

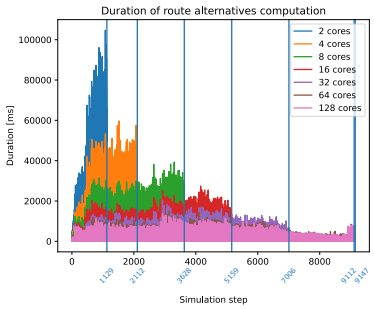

Figure 3. Cores Utilization

number of alternative routes, *number of samples* of Monte Carlo simulation performed by the PTDR algorithm, and a *near distance* which represent the view of a driver, i.e., after the near distance they need to do a forecast of what happens farther on their route by PTDR. For each variant a simulation was performed on a computational node of the Karolina cluster [14]. The results are summarized in Figure 2. The red line outlines the most suitable input parameters. The parameters above the line are worse in terms of quality or performance. We can also see a setting that is better in performance but little worse in quality.

5. Conclusion

Based on the simulator's parameter space exploration we have identified the most appropriate settings for the simulator. With this setting we have also performed a test of scalability. The results are shown in Fig. 3. The graph shows the scalability of route alternatives computation as this part is the most demanding part of algorithm. It can be seen that algorithm scales up to 64 cores. The slight difference between 128 and 64 cores is given by the number of active vehicles. The maximal number of active vehicles in one step is 1,269. With a higher number of active vehicles the algorithm most likely scales up. Moreover, we have identified the route alternatives algorithm as a good candidate to port into Rust to improve the performance.

Future work is focused on optimizing most demanding parts of the simulators with help of new accelerators such as FPGAs.

Acknowledgments

This work was supported by the EVEREST project - the European Union's Horizon 2020 research and innovation programme under grant agreement No. 957269 and the LIGATE project. LIGATE project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 956137. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Italy, Sweden, Austria, the Czech Republic, Switzerland. This project has received funding from the Ministry of Education, Youth and Sports of the Czech Republic (ID: MC2102). The work was also supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90140).

References

[1] C. Pilato, S. Bohm, F. Brocheton, *et al.*, "Everest: A design environment for extreme-scale big data analytics on heterogeneous platforms," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2021, pp. 1320–1325. DOI: 10.23919/DATE51398.2021.9473940.

[2] "Hyperqueue." (2022), [Online]. Available: https://github.com/it4innovations/hyperqueue (visited on 08/04/2022).

[3] M. Rocklin, "Dask: Parallel computation with blocked algorithms and task scheduling," Jan. 2015, pp. 126–132. DOI: 10.25080/Majora-7b98e3ed-013.

[4] P. Moritz, R. Nishihara, S. Wang, *et al.*, "Ray: A distributed framework for emerging ai applications," in *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'18, Carlsbad, CA, USA: USENIX Association, 2018, pp. 561–577, ISBN: 9781931971478.

[5] E. Tejedor, Y. Becerra, G. Alomar, *et al.*, "Pycompss: Parallel computational workflows in python," *International Journal of High Performance Computing Applications*, vol. 31, Aug. 2015. DOI: 10.1177/1094342015594678.

[6] Y. Babuji, A. Woodard, Z. Li, *et al.*, "Parsl: Pervasive parallel programming in python," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '19, Phoenix, AZ, USA: Association for Computing Machinery, 2019, pp. 25–36, ISBN: 9781450366700. DOI: 10.1145/3307681.3325400.

[7] P. Di Tommaso, M. Chatzou, E. W. Floden, P. Barja, E. Palumbo, and C. Notredame, "Nextflow enables reproducible computational workflows," *Nature Biotechnology*, vol. 35, pp. 316–319, Apr. 2017. DOI: 10.1038/nbt.3820.

[8] J. Köster and S. Rahmann, "Snakemake—a scalable bioinformatics workflow engine," *Bioinformatics*, vol. 28, no. 19, pp. 2520–2522, Aug. 2012, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bts480. eprint: https://academic.oup.com/bioinformatics/article-pdf/28/19/2520/819790/bts480.pdf.

[9] "Hyperloom." (2019), [Online]. Available: https://github.com/It4innovations/HyperLoom (visited on 08/04/2022).

[10] V. Cima, S. Böhm, J. Martinovič, *et al.*, "Hyperloom: A platform for defining and executing scientific pipelines in distributed environments," English, in *ACM International Conference Proceeding Series*, Cited By :9, 2018, pp. 1–6. [Online]. Available: www.scopus.com.

[11] "Hyperqueue: Overcoming limitations of hpc job managers." (2017), [Online]. Available: https://sc21.supercomputing.org/proceedings/tech_poster/tech_poster_pages/rpost104.html (visited on 08/01/2022).

[12] "European union, "navigation system software" innovation radar / discover great eu-funded innovations." (2019), [Online]. Available: https://www.innoradar.eu/innovation/34415 (visited on 08/04/2022).

[13] E. Vitali, D. Gadioli, G. Palermo, *et al.*, "An efficient monte carlo-based probabilistic time-dependent routing calculation targeting a server-side car navigation system," English, *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 1006–1019, 2021, Cited By :3. [Online]. Available: www.scopus.com.

[14] "The petascale system karolina: Documentation." (2021), [Online]. Available: https://docs.it4i.cz/karolina/hardware-overview/ (visited on 08/01/2022).