Dynamic computing resource allocation in online flood monitoring and prediction

S Kuchar¹,², M Podhoranyi¹, R Vavrik¹ and A Portero¹

¹ IT4Innovations, VSB - Technical University of Ostrava, 17. listopadu 15/2172, 708 33 Ostrava, Czech Republic

E-mail: stepan.kuchar@vsb.cz

Abstract. This paper presents tools and methodologies for dynamic allocation of high performance computing resources during operation of the Floreon+ online flood monitoring and prediction system. The resource allocation is done throughout the execution of supported simulations to meet the required service quality levels for system operation. It also ensures flexible reactions to changing weather and flood situations, as it is not economically feasible to operate online flood monitoring systems in the full performance mode during non-flood seasons. Different service quality levels are therefore described for different flooding scenarios, and the runtime manager controls them by allocating only minimal resources currently expected to meet the deadlines. Finally, an experiment covering all presented aspects of computing resource allocation in rainfall-runoff and Monte Carlo uncertainty simulation is performed for the area of the Moravian-Silesian region in the Czech Republic.

1. Introduction

Europe has suffered a rising number of all kinds of extreme flood events [7] from coastal and fluvial to pluvial and reservoir floods. These events do not have natural borders of influence, and therefore many scientists and researchers study them. Extreme flood events have increased rapidly in frequency and magnitude during last decades [4], and flood monitoring systems were thus developed in order to analyze and model the current flood situation and provide short-term prediction to support emergency decision processes related to the threat of a given coming flood. River flood monitoring and control requires measurement and notification of the water level, velocity and precipitation [10].

Several types of monitoring systems have been developed to provide information about prediction of extreme flood events and monitoring of their current behavior [13]. These systems provide similar functionality, but on different levels of complexity [3]. There are systems, which offer information only about the current state and mainly serve as dispatching systems, that coordinate usage of different kinds of human and material resources, preparing emergency plans and deploying experts and equipment. A different type of system can be more focused on prediction of the upcoming situation based on the current and historical situation as well as the meteorological forecast. These systems execute various hydrologic model implementations and analyze their results to provide flood warnings before the flooding situation escalates. As these systems often work automatically in near-real time, they need appropriate computing resources to provide accurate up-to-date results and timely warnings. Allocating these resources is not a trivial task because the executed models can be computationally intensive, and often the results must be computed quickly to be of any use for the emergency services that base their decisions on timely and precise information. The execution time of simulations during...
emergency scenarios should be in the order of tens of minutes, and the first warning should come at least several hours or even days before the emergency as far as the atmospheric forecast data are available. Lately, high performance computing (HPC) infrastructures and methods have piqued the interest of researchers and practitioners in the field of flood monitoring to increase the spatial and temporal scale and resolution of flood inundation models [16], improve the accuracy of modelling results with calibration [17], work with the probabilistic nature of the numerical weather prediction in ensemble flood forecasting [6], provide urgent computing in flood decision support [2] and tackle other computation intensive tasks while ensuring short execution times.

Nevertheless, effective flood monitoring systems are not only about providing the most precise results as soon as possible, but they also have to react to the seasonality of floods. In general, floods have seasonal character and there are periods that are not affected by this phenomenon [8]. This means that it is not economically effective to operate a flood monitoring system in the full performance mode during its whole lifetime (i.e., using a large amount of computing resources to provide results as quickly as possible even during non-emergency scenarios). On the other hand, it is important to keep the system running even in non-flood seasons with a low number of computing resources (and therefore a lower energy consumption) because the risk of flood is still present. Therefore, proper distribution of various kinds of resources based on the current flood situation may decrease cost of the operation.

In this paper, an HPC based online flood prediction system is presented that is able to dynamically allocate appropriate computational resources in hydrological modelling to increase the energy efficiency during non-flood situations and ensure more precise results with shorter execution time during emergencies.

2. Floreon$^+$ system
Floreon$^+$ is a system for monitoring, modeling, prediction and support in disaster management that was primarily developed for the Moravian-Silesian region in the Czech Republic. As it is a web-based platform backed by an HPC infrastructure, its modularity, flexibility and responsiveness allows easy integration of various thematic domains, regions and data. Its main goal is to support operational and tactical disaster management and emergency processes by providing and integrating simulations from several domains in near-real time.

The system currently focuses on the domains of flood monitoring and prediction, monitoring of the real time traffic situation and modeling of toxic cloud spread.

In the area of flood monitoring and prediction, Floreon$^+$ computes an ensemble of rainfall-runoff (RR) and hydrodynamic (HD) models. The system currently supports two semi-distributed event RR models – HEC-HMS 3.4 (developed by USACE) [14] and Math1D (experimental internally developed model) - and executes them both for all monitored catchments. The measured input data to the models are provided every 10 minutes from a network of water and meteorological gauges managed by the Odra catchment management office. A meteorological forecast for 72 hours is acquired every 6 hours from the numerical weather forecast model MEDARD provided by the Academy of Sciences of the Czech Republic. Results from the RR models are passed to the HD models that compute water levels, velocities and discharge, which are then visualized on the map interface. Currently, the only supported HD model in the system is HEC-RAS 4.1 (developed by USACE) [15], but we are planning to extend this support with additional models (focusing on 2D models, such as MIKE21FM by DHI).

Standard simulations of the supported RR models in the Floreon$^+$ system can also be enhanced by running automatized calibrations of their parameters that use optimization and inverse modelling methods. These methods estimate schematization parameter values that minimize the error between the measured and simulated values based on the current state of the catchment and meteorological information. Some input data cannot be calibrated due to its uncertain nature (e.g. weather forecasts), but this data significantly influences the accuracy of the results. Fortunately, it is possible to model the distribution of errors from the comparison of historical results with real measured values [5]. These distributions are then used by the Floreon$^+$ system to run Monte Carlo (MC) uncertainty simulations.
that provide a probabilistic view on possible scenarios in the near future. Both automatized calibration and uncertainty modelling execute a high number of simulations with changing parameters, so their computation time is long when executed sequentially. Parallelization and HPC infrastructure is used to decrease the execution time to meet the required quality of service. (Additional information about the complete online flood monitoring and prediction process in the Floreon+ system can be found in [9]).

In the domain of traffic monitoring, Floreon+ uses statistical methods to provide an overview of the current road traffic situation based on the velocity and location data from on-board GPS devices coming to the system every minute.

3D computational fluid dynamics solvers are also used in the system to simulate the extent and concentration of a possible toxic cloud spread for accidents occurring in places with known risks or contamination from mobile sources.

All outputs of the system are provided through a web interface with a map component (see Figure 1). This solution consolidates all of the integrated sections into one complex geographical framework and therefore delivers a detailed overview of the current situation as well as historical events. The combination of three simulation domains puts additional pressure on the computing resource management, sharing and balancing because available resources have to be shared according to the current priority in various domains (e.g., during a flood emergency, the majority of the computing resources should be used for flood simulations and not expended on toxic cloud modelling). Even though we currently focus only on resource allocation in the online flood monitoring process, the proposed methodology is also usable for other domains.

![Figure 1](image.png)

**Figure 1.** Screenshot of the Floreon+ web interface.

3. **Runtime resource management**

Applications running simulations in any domain have to satisfy certain runtime constraints. Generally, they need to provide defined precision of results in a given time limit. To ensure fulfilment of the constraints, when several applications run concurrently and share resources of the same machine, some sort of automatic resource management has to be applied. In our system, the HARPA-OS runtime resource manager [1], which is the topmost layer of the HARPA platform [12], is used for dynamic resource allocation. It is able to monitor the status of system resources and assign or reallocate resources between all managed applications during their execution. The main aim of the resource allocation strategy is to provide a sufficient amount of resources for the application to fulfill its constraints, but the objective is also to keep power consumption as low as possible and save energy.
All managed applications have to implement a specific interface, which enables runtime reconfiguration of the application in cycles and contains the following methods:

- **onSetup()** – this method is executed only once at the start of the application. It contains initialization of the application, loading of input data and configuration of parameters as well as necessary pre-processing. For example, for RR modelling, a number of MC samples is set based on the expected precision, and an initial simulation without uncertainty is executed to provide referential results. For HD modelling, a time step for the HD simulation is set and appropriate data are extracted from input hydrographs.

- **onConfigure(int awmId)** – this method is executed when the HARPA-OS assigns resources to the application for the first time or changes the allocated resources to meet the expected service quality. HARPA-OS selects the most appropriate set of resources (passed to the method by the awmId parameter; see section 4. for more information) based on the current scenario, resource allocation strategy, state of the application and required service quality, and passes it to the onConfigure method. The application then has to reconfigure its parameters to properly run on the allocated resources. RR models reconfigure the number of MC samples processed in the current cycle. HD models change the number of time steps executed concurrently in the current cycle (a steady flow approach is used for HD modelling, so each time step of the simulation is independent and can be executed in parallel).

- **onRun()** – this method performs the main computation of the application. The configured number of MC samples and HD time step simulations for the current cycle is processed, and the average time required to simulate one sample or time step is measured.

- **onMonitor()** – this method monitors the current performance and evaluates the ongoing progress of the application execution and its constraints. After all MC samples and HD time steps in the current cycle are processed, the time required to process the remaining samples/time steps is computed, and a total time for the whole simulation is estimated. If the estimated time would exceed the time constraint specified in the service quality level, more resources are requested.

- **onRelease()** – this method performs all required post-processing steps and cleanup operations after the whole computation is finished. In RR modelling, percentiles are extracted from the MC sample results and saved as final results. In HD modelling, the results are combined with a digital elevation model of the area to extract the inundated areas.

Figure 2 illustrates a simplified example for execution of three independently managed applications running concurrently on the same machine and sharing its 16 CPU cores. These can represent several uncertainty modelling instances, each processing a different modelled area or a combination of uncertainty and hydrodynamic simulations. The first application has a high priority and a short execution time limit (e.g. a flood emergency occurring in the processed catchment), so it is clearly preferred in the first cycle with 8 core allocation, but needs only 6 cores in the second cycle to finish its computation. The resource manager can therefore reallocate the 2 released cores to the second application, while resources of the third application remain unchanged (note that no onConfigure method is called in the third application as its resources did not change). Finally, 6 cores released by the end of the first application are reallocated to the remaining applications in the third cycle. The resource manager compares the monitoring results from their onMonitor methods and decides to allocate 4 cores to the third application and the 2 remaining cores to the second application.

In real simulations, the duration of each cycle depends on the implementation, but it should be short enough to provide enough possibilities to re-allocate the resources during the whole runtime. In our implementation of RR uncertainty modelling, each cycle is just a few seconds long, and for HD simulations, it is in the order of tens of seconds. This means that during a standard 10-minute RR uncertainty simulation execution, the re-allocation can occur hundreds of times. The re-allocation can therefore be highly dynamic and it can quickly adapt to the changing load of the computing resources and required service quality levels of all running applications.
4. Quality of provided services

All allowed resource configurations (e.g. number of allocated x86-64 cores and cluster nodes, amount of required memory, allocation of accelerators, etc.) that the managed application is able to work with have to be pre-specified by a set of application working modes (AWM). The applications also have to set their quality of service (QoS) levels that specify their runtime requirements, such as the priority of each task or its requested execution time limit. The HARPA-OS then allocates computing resources to each running task based on the best fitting AWM for the current quality expectations in each cycle.

We have identified three major QoS levels during the Floreon+ system operation that are based on the requirements of the system users (integrated emergency center in the Moravian-Silesian region and expert users) and periodicity of the input data coming to the system from third party providers. The switching between these QoS levels is triggered by exceeding specific flood activity degrees (FAD). FAD describes the state of the water level during flood events. Usually, there are three levels of FAD quantified by a water level and defined for all major water gauges in the Czech Republic; the first degree is alertness, the second degree is preparedness and the third degree is flood hazard. These flood activity degrees are based on the local water and river channel conditions.

The QoS levels used in our system are:

- **Standard operation** – a standard scenario for non-flood situations. The computation can be relaxed with lower precision, and the system should only use a minimal number of resources for flood monitoring. Rainfall-runoff simulations are executed every hour along with their uncertainty simulations generating 1000 MC samples. Hydrodynamic simulations are started once a day with a 6-hour execution time.
- **Alertness** – when measured or simulated discharge on any station exceeds the first flood activity degree, the system switches to an alerted mode. Simulations are started more often with a higher focus on precision to more carefully monitor the development in rivers. Rainfall-
runoff simulations are executed every 30 minutes with a 10 minute time step, along with their uncertainty simulations generating different numbers for different modelled catchments – 1250 samples for the Opava catchment, 2000 for the Odra (upper part) catchment, 2500 for the Olše catchment and 3000 for the Ostravice catchment (the modelled catchments are described in section 5.1). Hydrodynamic simulations are started every 6 hours with a 2-hour execution time.

- Emergency – when the measured or simulated discharge on any station exceeds the second flood activity degree, the system switches to an emergency mode. Excess resources are allocated to provide the simulation results as soon as possible with high precision demands, and the flood monitoring domain has the highest priority. Rainfall-runoff simulations are executed every 10 minutes, along with their uncertainty simulations, generating 3000 MC samples. Hydrodynamic simulations are started every 30 minutes with a 30-minute execution time.

Application working modes have to be specified separately for each type of simulation (i.e. one set of AWMs for simple rainfall-runoff, different set for uncertainties and another one for hydrodynamics) to support the identified QoS. One AWM for each type has to cover the maximum allocable resources needed to support the Emergency mode. One AWM has to contain only minimal resources for economical operation in standard mode. Several AWMs have to exist in between to ensure proper dynamic reallocation of resources during execution when simulation is delayed because of more prioritized requests or errors in the infrastructure.

5. Experiment
To show the functionality of the HARPA-OS runtime resource manager, we have conducted an experiment that executes and manages rainfall-runoff simulations and their uncertainty simulations for the modelled area of the Floreon system.

5.1. Area of interest
The Moravian-Silesian region is situated in the northeastern part of the Czech Republic in central Europe on the borders with Poland and Slovakia. The western and eastern parts of the region are mountainous. Hydrologically, the territory of the Moravian-Silesian region lies within the river Odra and Morava catchments. The Floreon system currently only covers the Odra catchment, which can be divided into four delineated sub-catchments – the Ostravice, Olše (excluding the part of the catchment in Poland) and Opava catchments and the upper part of the Odra catchment up to the confluence with the Opava river (see Figure 3). Basic characteristics of the modeled catchments are depicted in Table 1.

5.2. Experimental infrastructure
The experiment has been conducted on a machine with a SuperMicro X10DAI Motherboard with two sockets. Each socket arms an Intel(R) with 8 cores Core(TM) i7-4870HQ CPU @ 2.50GHz, providing 16 cores in total, as well as 32 threads with hyper-threading that is disabled for these experiments. The machine is equipped with 256 GB DDR3 1866 MHz without ECC memory.

5.3. Runtime resource management settings
To retain large flexibility and show even small changes in resource allocations, we opted to define finely grained AWMs in this experiment. In total, there are 16 AWMs starting with one CPU core and going through by one core to the whole machine. Numbers of Monte Carlo samples and related execution time limits were set to the QoS levels identified in section 4.
Table 1. Basic characteristics of studied catchments (in order of importance).

| Catchment      | Area (km²) | Length (km) | Spring (m.s.l.) | Related risks                                                                                                                                 |
|----------------|------------|-------------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| Ostravice      | 828        | 64          | 720             | High population density, Industrialized zones, Major cities included, Mountains in the upper part can cause significant runoff                 |
| Olše           | 914        | 72.8        | 909             | Probability of fluvial floods, Narrow channel, Cross border catchment, Problematic monitoring of Polish tributaries                         |
| Odra (upper part) | 1615      | 112         | 633             | Flash flood occurrence, High density of streams in the upper part, No dams, Missing water stations at tributaries                           |
| Opava          | 2089       | 110.7       | 529*            | Dense river network, Extensive area, Lack of water monitoring stations, Agricultural areas                                                   |

* confluence of Bíla, Střední and Černá Opava
5.4. Results

Based on the four modelled catchments, four instances of rainfall-runoff and uncertainty simulations were started on the experimental infrastructure at the same time and had to compete for resources. We performed three experimental runs, one for each identified emergency level. All instances started with the same amount of resources, as the resource manager divides all resources in the machine equally.

Figure 4 shows the Standard operation mode where the resource manager quickly decreases equally divided allocated resources of all instances as it receives information that the simulations do not need so much computing power to meet their specified execution time limits. All instances are then kept at the lowest possible resource levels until the end of the computation, leaving the remaining resources free to be used by other applications. The gap with ideal performance grows exponentially, as one core is more than enough to compute simulations in this mode.

Figure 5 shows the Alertness mode, in which the Opava (top right) catchment has the least MC samples to compute. Therefore, it only needs minimal resources and even finishes before its time limit to release its allocated core for use by other catchments. The Odra catchment (top left) has the second lowest priority and MC sample count, but it already needs more than one core to finish its computation in the specified time limit for the Alertness mode. The results show how it switches between a 2 and 3-core allocation during its execution as the total execution time estimate reacts to the current allocation. The Olše catchment with the second highest importance (bottom right) has an even higher allocation of 3 or 4 cores and even 5 near the end of the simulation. Allocated resources also increase near the end of the simulation for the most important Ostravice catchment (bottom left) that shows the most dynamic behavior. This is caused by the fact that the total execution time estimate is becoming more accurate near the end and the resource manager tries to finish the simulation right on time.

Figure 6 shows the Emergency mode, where all simulations finally fill the entire machine and the resource manager should pause all other non-flood applications or decrease their resources to the

---

**Figure 4.** Experimental results for the Standard operation mode.

**Figure 5.** Experimental results for the Alertness operation mode.

**Figure 6.** Experimental results for the Emergency operation mode.
absolute minimum. All instances balance on a 4-core allocation, competing only when any of them become significantly delayed.

**Figure 5.** Experimental results for the Alertness mode.

**Figure 6.** Experimental results for the Emergency mode.
6. Conclusion and future work

The main aim of the Floreon project is to provide detailed information about the possibility of any incoming flood and its extent. In the future, Floreon should be able to estimate possible property damage before it happens, provide recommendations about flood measures and online decision support for emergency committees to minimize the number of casualties and property damage in the case of extreme weather events. HARPA-OS integration will significantly increase the quality of the provided services, mainly by being able to adapt the computation demand to the changing environment and weather situation, conserving energy and costs of continuous operation and ensuring availability of the system.

The main advantages that HARPA-OS integration can bring to online flood monitoring systems are:

- More cost-efficient continuous operation in non-emergency scenarios, heightened alertness and prioritized execution in warning scenarios, and urgent computation in emergency scenarios
- Optimal reallocation of computational resources during runtime for concurrent execution of several different simulations with different service quality levels
- Faster reactions to errors in the infrastructure – the runtime resource manager can easily reallocate computation from the faulty components

The experiment in this paper focused only on a single-node operation, but to effectively utilize HPC resources, a multi-node approach has to be covered as well. We are currently developing an experimental version of an MPI-based solution that will be able to manage resources over multiple computing nodes. Our future research will also focus on enhancing the proposed methodology by adding priorities for individual instances that can dynamically change during execution based on the state of the catchment and meteorological forecast to better react to local changes (e.g. local flash floods). We also plan to extend the methodology further to other problem domains in the Floreon system.

Acknowledgement

This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project “IT4Innovations excellence in science - LQ1602”, by the Large Infrastructures for Research, Experimental Development and Innovations project “IT4Innovations National Supercomputing Center – LM2015070” and by the 7th EU framework programme project no. FP7-612069 HARPA.

References

[1] Bellasi P, Massari G and Fornaciari W 2015 Effective runtime resource management using Linux Control Groups with the BarbequeRTRM Framework ACM Trans. Embed. Comput. Syst. 14:2 pp 39:1–39:17
[2] Brzoza-Woch R, Konieczny M, Kwolek B, Nawrocki P, Szydlo T and Zielinski K 2015 Holistic approach to urgent computing for flood decision support Procedia Computer Science 51 pp 2387–96
[3] Cools J, Innocenti D and O’Brien S 2016 Lessons from flood early warning systems Environmental Science and Policy 58 pp 117-22
[4] Dobrovicova S, Dobrovic R and Dobrovic J 2015 The Economic impact of floods and their importance in different regions of the world with emphasis on Europe Procedia Economics and Finance 34 pp 649–55
[5] Golasowski M, Litschmannova M, Kuchar S, Podhoranyi M and Martinovic J 2015 Uncertainty modelling in rainfall-runoff simulations based on Parallel Monte Carlo Method Neural Network World 25:3 pp 267–86
[6] Haiyun S, Li T, Liu R, Chen J, Li J, Zhang A and Wang G 2015 A service-oriented architecture for ensemble flood forecast from numerical weather prediction Journal of Hydrology 527 pp 933–42
Kjeldsen T R, MacDonald N, Lang M, Mediero L et al. 2014 Documentary evidence of past floods in Europe and their utility in flood frequency estimation. *Journal of Hydrology* **517** pp 963-73

Koutroulis A G, Tsanis I K and Daliakopoulos I N 2010 Seasonality of floods and their hydrometeorologic characteristics in the island of Crete. *Journal of Hydrology* **394** pp 90-100

Kuchar S, Golasowski M, Vavrik R, Podhoranyi M, Sir B and Martinovic J 2015 Using high performance computing for online flood monitoring and prediction. *International Journal of Environmental, Chemical, Ecological, Geological and Geophysical Engineering* **9**:5 pp 11-16

Merkuyeva G, Merkuyev Y, Sokolov B V, Potryasaev S, Zelentsov V A and Lektauers 2015 Advanced river flood monitoring, modelling and forecasting. *Journal of Computational Science* **10** pp 77-85

Portero A, Kuchar S, Vavrik R, Golasowski M and Vondrak V 2014 System and application scenarios for disaster management processes, the rainfall-runoff model case study. *Lecture Notes in Computer Science* **8838** pp 315–26

Rodopoulos D, Corbetta S, Massari G, Libutti S, Catthoor H, Sazeidesz Y, Nicopoulos C, Portero A, Cappe E, Vavrik R, Vondrak V, Soudris D, Sassi F, Fritsch A and Fornaciari W 2015 HARPA: Solutions for dependable performance under physically induced performance variability. *Proceedings of International Conference on Embedded Computer Systems: Architectures, MOdeling, and Simulation SAMOS XV Conference* July 20-23, 2015.

Sunkpho J and Ootamakorn Ch 2011 Real-time flood monitoring and warning system. *J. Sci.Technol.* **33** pp 227–35

USACE 2009 *US Army Corps of Engineers, Hydrologic Engineering Center, HEC-HMS, Hydrologic Modeling System HEC-HMS. User’s manual, version 3.4*

USACE 2010 *US Army Corps of Engineers., Hydrologic Engineering Center, HEC-RAS, River Analysis System. Hydraulic Reference manual, version 4.1*

Yu D 2010 Parallelization of a two-dimensional flood inundation model based on domain decomposition. *Environmental Modelling & Software* **25**:8 pp 935–45

Zhang A, Li T, Si Y, Liu R, Shi H, Li X, Li J and Wu X 2016 Double-layer parallelization for hydrological model calibration on HPC systems. *Journal of Hydrology* **535** pp 33–45