Demo: Server-Assisted Interactive Mobile Simulations for Pervasive Applications

Christoph Dibak, Frank Dürr, Kurt Rothermel
Institute of Parallel and Distributed Systems
University of Stuttgart, Stuttgart, Germany
Email: {dibak, duerr, rothermel}@ipvs.uni-stuttgart.de

***Abstract*—Emerging augmented reality devices allow for visualizing results of numerical simulations ubiquitously. This enables decision makers and engineers in the field to make better decisions. However, computation of resource-intensive simulation models on resource-poor and battery-powered mobile devices requires to drastically reduce the quality of the simulation. We therefore proposed approaches utilizing a remote server and the Reduced Basis Method (RBM) to generate a reduced model of the simulation.**

In our demo, we demonstrate the effectiveness of our approaches for mobile simulations. Our demo consists of two devices, a mobile device and a server. Both devices are connected via wireless network. The mobile device visualizes the result of a complex computation. Parameters of the simulation are effected by sensors of the mobile device. The user can choose between computing on the mobile device, computing solely on the server, or computing using our approach.

***Index Terms*—Mobile Computing ; Mobile Cyber-Physical Systems; Mobile Simulations**

I. Introduction

Emerging augmented reality (AR) devices, such as the Microsoft Hololens, allow for new forms of interaction with software. One interesting use-case for AR is to display the result of numerical simulations [1]. This helps decision makers and engineers to make decisions in the field [2], [3]. For example, consider an engineer who has to place a hot exhaust tube in a factory according to surrounding materials. Using AR, the engineer can see the heat of the exhaust tube as if the machine were operational. The engineer can adjust parameters of the simulation to change different operation modes of the machine or the airflow in the factory and check if surrounding material withstands the temperature.

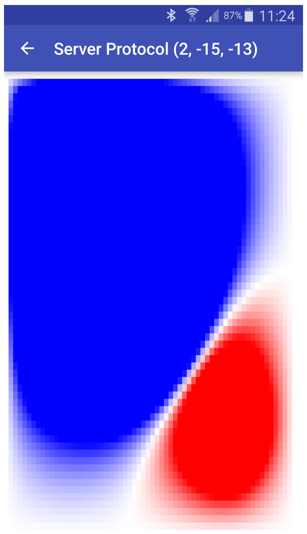
Typically, the system to be simulated is described by means of differential equations. These equations can be discretized, e.g., using finite differences or finite elements. Discretization leads to a system of algebraic equations. Solving these equations provides a vector as the simulation result. Usually, the model equation contains parameters that can be changed prior to runtime of the simulation. A user wearing an AR headset might want to change the parameters and see the effect on the simulation immediately.

While problems of AR, such as head-movement tracking and display of 3D scenes to the user, are solved in hardware, the main challenge remains to compute the numerical simulation on the mobile device. AR devices are battery-powered and therefore processors need to be energy efficient. Additionally, the computation should have low latency such that the user can interactively change parameters and see results of the simulation in real-time. All these requirements cannot be met by the mobile device itself. In our approach, presented in a full paper at PerCom 2017 [1], we therefore, on the one hand, utilize computing resources of a remote server, and, on the other hand, use a reduced model of the simulation generated by the Reduced Basis Method (RBM).

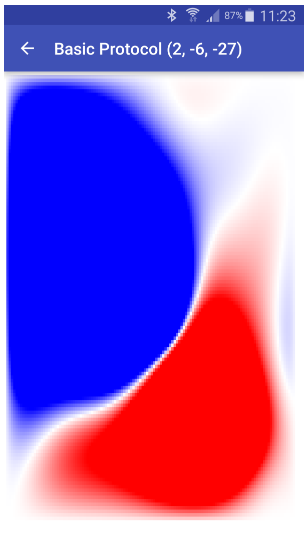
The RBM provides an approximation of the simulation result as linear combination of snapshots [4]. Snapshots are pre-computed results of specific parameters of the simulation. The parameters of the snapshots are selected in a basis generation process. During runtime, a pre-computed problem matrix needs to be solved that is typically much smaller as the problem matrix of the full simulation problem. As pre-condition for applying the RBM, the problem matrix of the algebraic problem needs to fulfill parameter separability. Parameter separability means that most of the problem does not depend on the parameters and can therefore be pre-computed in the basis generation process.

Implementing a mobile simulation system requires the involvement of both, simulation experts implementing the simulation, and app programmers implementing the mobile app. Typically, both parties use different existing libraries, implementation languages, and platforms. Simulation experts typically use existing numerical libraries for x86. App programmers use – if targeting the popular Android platform – Java and ARM architectures. To facilitate the system development, our middleware provides convenient interfaces between the code of the simulation expert running on the server and the code from the app developer running on the mobile device.

In this demo, we present an app running on a mobile device that computes the simulation result of a system modeled by the diffusion-advection equation. This is a general equation which can be used, e.g., to simulate the distribution of heat in solid objects. The equation has three parameters, one for the diffusion and two for advection in x and y direction. We use the accelerometer and the magnetic field sensor of the Smartphone as input to the simulation, such that advection terms in x and y direction change when the attitude of the device is changed. The result of the simulation is visualized on the mobile device. Figure 1 shows screenshots of the demo app showing the simulation result. Results can be interpreted as heat distribution on the surface of an object. In



(a) low resolution result from the server



(b) high resolution result using our approach

Fig. 1: Screenshots of the app.

the visualization, we see one heat sink and one heat source. Depending on the attitude of the device, the heat distribution changes.

In addition to our approach, we also implemented an approach where the simulation is computed solely on the server and an approach where the simulation is computed solely on the mobile device. The demo shows that our approach is much faster and provides better quality for a target frame rate.

II. System Overview

In this section, we give an overview of the hardware and software components before we will provide some details about the implementation in the next section. Figure 2 depicts an overview of the two hardware components, namely the server and the mobile device, and the four software components, namely the simulation component, the middleware part running on the mobile device, the middleware running on the server, and the application using the simulation results.

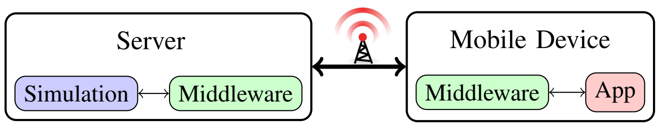


Fig. 2: server, mobile device, and software components

Our approach uses the Reduced Basis Method (RBM) to generate a reduced basis on the server. It then sends the reduced basis to the mobile device. Once the reduced basis is available on the mobile device, our approach handles queries from the app by solving the reduced problem.

A. Hardware Components

Our system consists of a remote server located, for instance, in a data center, and a Samsung Galaxy Note 4 as Smartphone. The two devices are connected over the Internet via IEEE 802.11 (WiFi) for the last hop to the mobile device.

B. Software Components

The system consists of three software components, namely the simulation code, the distributed middleware, and the app.

The simulation code computes the result of the diffusion advection equation, which is given as

$$\mu_{diff}\Delta u + \mu_{advx}\partial_x u + \mu_{advy}\partial_y u = f. \qquad (1)$$

This equation can be discretized using finite differences to obtain an algebraic equation $A(\mu)u(\mu) = f(\mu)$, where $A(\mu)$ is a given matrix, $f(\mu)$ is a given vector, and $u(\mu)$ is the unknown solution for the problem. The parameter vector μ includes values for μ_{diff}, μ_{advx}, and μ_{advy}.

In addition to the simulation results, the simulation code also provides a parameter separable representation of matrix $A(\mu)$ in the form

$$A(\mu) = \sum_i \theta_i(\mu) A_i, \qquad (2)$$

where A_i are matrices in the same shape as $A(\mu)$, and θ_i are functions mapping from the parameter to scalar real values. This representation can be obtained directly from the discretization of Equation 1. The right-hand side of Equation 1 does not depend on any parameter.

The middleware provides simulation results to the app. It requests full simulation results from the simulation code in order to construct a reduced model of the simulation. This reduced model will be sent to the mobile device and is used to answer queries of the mobile app.

The app running on the mobile device reads sensor data from the mobile device, requests simulation results from the middleware, and visualizes the results on the screen. Sensor data is read from the orientation sensors of the device and used as input to the diffusion-advection equation. Parameters change with the attitude of the mobile device, e.g., the parameter for the advection in x and y direction is changed when the Smartphone is tilt right-left or top-down. The parameter for the diffusion changes based on readings of the accelerometer.

III. Implementation

In this section, we describe some details of the implementation. We split the description into three parts, the numerical simulation implemented by the simulation expert, the middleware, and the mobile application implemented by the application programmer.

A. Simulation Implemented by the Domain Expert

We implemented the solver for the diffusion-advection equation in Python using the popular NumPy and SciPy libraries. These libraries provide optimizations for the x86 architecture of the server.

B. Middleware

We describe the basis generation on the server, the communication between server and mobile device, and the mobile part of the middleware.

a) Server Part of the Middleware: The server part of the middleware implemented in Python serves requests from the mobile part of the middleware running on the mobile device. For the basic approach, it serves two types of requests: file requests and basis generation requests. Both requests return reduced bases. For file requests, the mobile part of the middleware specifies the name of the file. For basis generation requests, it specifies the training set of parameters and the maximum residual.

When a basis generation request is received at the server, the middleware will create a new basis using the greedy approach [5]. In this iterative basic generation process, the middleware will choose the parameter of the training set that indicates the highest error, say $\mu_{\max}$. As error indicator, we use the residual of the equation, which has a well-known semantic for the simulation expert and has an efficient implementation using pre-computing (c.f. Appendix in our full paper [1]). After finding parameter $\mu_{\max}$, the middleware will call the simulation code with $\mu_{\max}$ as parameter to obtain a new snapshot for the reduced basis. Once the quality threshold given by the mobile device is reached, the basis generation process stops and the reduced basis is sent to the mobile device.

b) Communication: The generated reduced basis is communicated to the mobile device using the Msgpack library. The data structure consists of the snapshots and the pre-computed values for the reduced problem, namely the right-hand side, and matrices for the residual computation. These matrices and vectors are communicated in parameter-separated form as sums of theta functions and matrices.

Theta functions in the separation are serialized as strings. The only operation on theta functions for the RBM is multiplication, which is implemented as string concatenation. All theta functions can be represented as product of *basic theta functions*. Basic theta functions are the theta functions in separation of the problem. For our implementation, we only had 5 basic theta functions.

c) Mobile Part of the Middleware: The mobile part of the middleware stores pre-computed data on the mobile device and solves the reduced problem using the reduced basis. It is implemented in Java as Android library and can be included into apps programmed by the application programmer.

Once communicated from the server, the pre-computed data is stored on the internal storage of the mobile device. During runtime, the reduced problem is read from internal storage into memory. As the reduced problem is very small, it can be computed very fast, even without hardware optimization, using the popular Apache Commons Math library.

C. Mobile Application using the Middleware

Our middleware provides an easy-to-use abstraction of the simulation for the application programmer. The programmer only needs to call one method that will return the simulation result as vector. Therefore, the application programmer can focus on sensor input and visualization for the application. For the demo, sensor input is taken from the orientation sensors of the mobile device.

IV. Conclusion

We briefly presented our demo setup for our mobile simulation approach utilizing the Reduced Basis Method (RBM) for distributed mobile simulations. Our setup consists of a remote server and a mobile device. We utilized the RBM such that the mobile device can approximate the solution of the full simulation problem very fast and efficiently.

The demo shows the result of the diffusion-advection equation based on parameters. To demonstrate the performance with live sensor data, the parameters of the simulation are changed based on the orientation sensors of the mobile device. The parameters for advection are changed when the Smartphone is tilt right-left or top-bottom. The parameter for the diffusion is changed when moving the Smartphone up and down.

The demonstrator shows that using the RBM is much faster than computation of the full model on the mobile device or computation on a remote server and sending everything to the mobile device. We also implemented two different modes, where in one mode, the dimension of the underlying full problem is adapted depending on the execution time of the simulation model. Depending on the hardware specification and the available bandwidth to the server, our approaches are multiple quality-levels better and provide faster and more efficient results.

For further details on the methods implemented for this demo, we refer to our full paper at PerCom 2017 [1].

Acknowledgments

The authors would like to thank Anshuman Dash for his help in the implementation. Furthermore, the authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/2) at the University of Stuttgart.

References

[1] C. Dibak, A. Schmidt, F. Dürr, B. Haasdonk, and K. Rothermel, "Server-assisted interactive mobile simulations for pervasive applications," in *Proceedings of the 15th IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Mar 2017.

[2] C. Dibak and B. Koldehofe, "Towards Quality-aware Simulations on Mobile Devices," in *Proceedings of the 44. Jahrestagung der Gesellschaft für Informatik e.V. (INFORMATIK)*. Gesellschaft für Informatik (GI), Sep 2014.

[3] C. Dibak, F. Dürr, and K. Rothermel, "Numerical Analysis of Complex Physical Systems on Networked Mobile Devices," in *Proceedings of the 12th IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS)*, Oct 2015.

[4] B. Haasdonk, "Reduced basis methods for parametrized pdes – a tutorial introduction for stationary and instationary problems," Tech. Rep., 2014, chapter to appear in P. Benner, A. Cohen, M. Ohlberger and K. Willcox: "Model Reduction and Approximation for Complex Systems", SIAM.

[5] K. Veroy, C. Prud'Homme, D. V. Rovas, and A. T. Patera, "A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations," in *Proceedings of the 16th AIAA computational fluid dynamics conference*, vol. 3847. Orlando, Florida, 2003, pp. 23–26.