# Performance improvement with parallel numerical model simulations in the field of urban water management

Michael Mair, Robert Sitzenfrei, Manfred Kleidorfer and Wolfgang Rauch

## ABSTRACT

Numerical models are used to enhance the understanding of the behavior of real world systems. With increasing complexity of numerical models and their applications, there is a need of more computational power. State of the art processors contain many cores on one single chip. As such, new programming techniques are required if all these cores are to be utilized during model simulation runs. This manuscript reviews the runtime and speedup behavior of parallel model analysis software (e.g. Calimero and Achilles) applied to simulation tools for urban water management (e.g. CityDrain3, EPANET2, SWMM5, par-SWMM). The potential of using a parallel programming environment for 'coordinating' tasks of multiple runs of commonly used modeling software is analyzed. This is especially interesting as the modeling software itself can be implanted sequentially or parallel. Performance tests are performed on a set of real-world case studies. Additionally, a benchmark set of 2,280 virtual case studies is used to investigate performance improvement in relation to the size of the system. It was found that speedup depends on the system size and the time spent in critical code sections with increasing number of used cores. Applying parallelism only at the level of the model analysis software performs best.

**Key words** | Achilles, Calimero, nested parallelism, runtime, speedup, virtual test set application

**Michael Mair** (corresponding author)
**Robert Sitzenfrei**
**Manfred Kleidorfer**
**Wolfgang Rauch**
Unit of Environmental Engineering,
Institute of Infrastructure,
University of Innsbruck,
Technikerstr. 13,
6020 Innsbruck,
Austria
E-mail: *michael.mair@uibk.ac.at*

## INTRODUCTION

Numerical models are applied in many research fields to better understand the behavior of real world systems. In the field of urban water management there are a number of software products frequently applied (e.g. EPANET, SWMM5, CityDrain3 (Burger *et al.* 2010)). Increasing complexity of models (i.e. increasing number of parameters), as well as modern analysis methods applied to urban water management, require a multitude of model simulation runs (e.g. in sensitivity-, scenario- or uncertainty analysis). As a result, more computational power is needed. In the last decades, the ever increasing demand for computational power was satisfied simply by increasing the clock frequency of Central Processing Units (CPUs). However, a point has been reached where this method of improving hardware performance is no longer efficient. Adding more CPUs on a single chip and leaving the clock frequency nearly unchanged was deemed a better alternative (Olukotun *et al.* 1996). This decision has consequently changed conventional programming techniques that developers are used to (Sutter 2005). The era of parallel programming has now reached all fields of software development as the necessary hardware has become available on inexpensive desktop machines (Hill & Marty 2008). This new trend implies several new programming paradigms and frameworks for concurrent programming on different levels. For example, fine-grained parallelism can be realized with the help of GPGPUs (general-purpose graphics processing units) or using SIMD (single instruction multiple data) registers of CPUs by exploiting data parallelism. Medium-grained parallelism is, for example, using programming paradigms like OpenMP (Dagum & Menon 1998) on shared memory systems and MPI (Message Passing Interface) (Gabriel *et al.* 2004) on distributed memory systems. On this level, the parallelization is mostly realized on functional concurrency. The third and last level of parallelism is called

doi: 10.2166/hydro.2013.287

coarse-grained parallelism where parallelism is on the level of the workflow within a scientific domain.

Numerical models in the field of urban water management are often implemented sequentially, e.g. SWMM5 (Rossman 2010) and EPANET2 (Rossman *et al.* 2000). Such algorithms can only utilize one single CPU within a simulation run. Recently developed or optimized numerical models, however, also contain parallel code e.g. CityDrain3 (Burger *et al.* 2010), par-SWMM (Burger & Rauch 2012) a parallel version of SWMM5. Moreover, work was carried out on exemplifying the potential to solve hydraulic network equations on GPGPUs (Crous *et al.* 2012), development of a parallel demand driven hydraulic solver by exploiting SIMD registers (Guidolin *et al.* 2013) and executing integrated flood models on clusters (Moya *et al.* 2013). The parallelization of certain algorithms within one software product (e.g. CityDrain3 and parSWMM) represents one of many solutions for decreasing runtime of model runs. For many analyzing techniques, numerous independent and dependent model simulation runs are required. This is the case, for example, during the model development process where one major step is to calibrate and validate model simulation results with measured data by adapting model parameters. The adaptation of the parameters is influenced by the deviation of previous results of model simulations and real world data (Kleidorfer *et al.* 2009). Alternatively, sensitivity analysis can be used to understand the behavior of a real world system. Mair *et al.* (2012a) or Möderl *et al.* (2011a) performed such analyses (spatial sensitivity analyses) to assess the vulnerability of water supply and sewer networks, respectively. By varying model parameters (e.g. simulating a conduit collapse or pipe burst) the consequences can be analyzed by observing the change of infrastructure-specific hydraulic performance. To get a complete analysis, all conduits or pipes have to be tested. Each model modification defines a new model setup, which has to be simulated. This also results in a huge amount of independent simulation runs.

The tasks described above (calibration, sensitivity analysis, uncertainty analysis, etc. of numerical models) can either be performed manually (by starting a sequence of model runs) or by a software in itself that coordinates these tasks. This manuscript shows the potential of using a parallel programming environment in such a 'coordinating' software for multiple runs of parallel and sequentially implemented modeling software in the field of urban water management. Usually, for such tests, a limited number of case studies can be used due restricted availability of case study data. However, evaluating the speedup behavior with one or more case studies results in case specific results which cannot (or at least are difficult to) be generalized or transferred to boundary conditions (e.g. memory usage). Thus, the aim of this study is to generate case unspecific results for the question at hand. Therefore, the impact of different model sizes on the overall runtime is analyzed with a benchmark set of 2,280 synthetic model setups (Möderl *et al.* 2011b) and additionally with three real model setups. With the presented approach, this knowledge gap can be addressed. Additional to Mair *et al.* (2012b) one more test case is analyzed where nested-parallelism is applied by using a parallel version of SWMM5 (Burger & Rauch 2012).

## MATERIAL AND METHODS

The analysis of urban water management approaches and techniques requires a large number of model simulations – in essence parameter variations within an initial model setup. Two main levels can be identified where a parallelization of programming code may decrease the overall runtime of the analysis.

1. The first level is to parallelize the programming code at the level of the modeling software (Figure 1, blank box). This has been done, for example, in the modeling software CityDrain3 (CD3) (Burger *et al.* 2010) or par-SWMM (Burger & Rauch 2012) (medium- or fine-grained parallelism). Consequently the runtime of a single model simulation is decreased with the help of several parallelization strategies. We define this level as MS-1 level (**M**odeling **S**oftware at level **1**).
2. The second level parallelization is done in the model analysis software (i.e. the 'coordinating' software, Figure 1, gray colored box) by executing model simulation processes in parallel. If we define the execution of a model simulation as a 'Task', this can be denoted as task level parallelism (coarse-grained parallelism). Consequently, there is no speedup for a single model

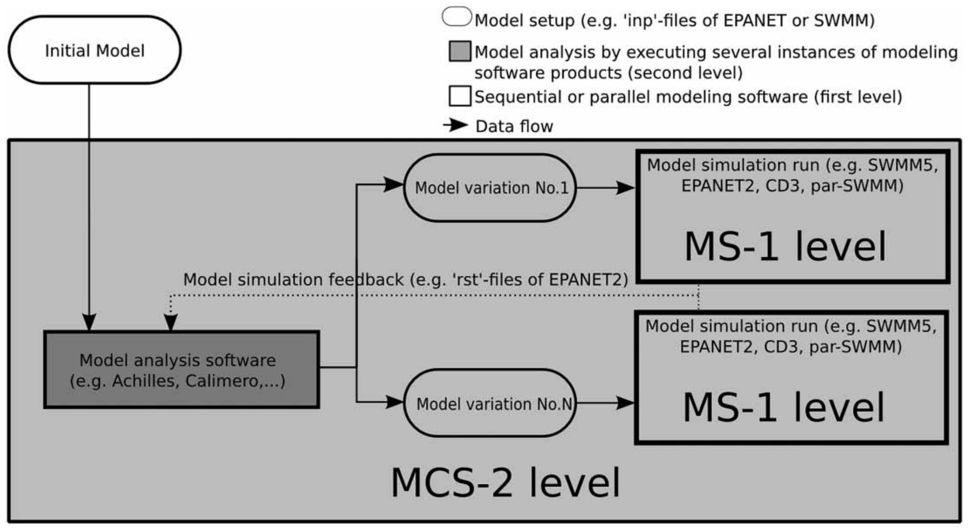
Model setup (e.g. 'inp'-files of EPANET or SWMM)
Model analysis by executing several instances of modeling software products (second level)
Sequential or parallel modeling software (first level)
Data flow
Initial Model
Model variation No.1
Model simulation run (e.g. SWMM5, EPANET2, CD3, par-SWMM)
MS-1 level
Model simulation feedback (e.g. 'rst'-files of EPANET2)
Model analysis software (e.g. Achilles, Calimero,...)
Model variation No.N
Model simulation run (e.g. SWMM5, EPANET2, CD3, par-SWMM)
MS-1 level
MCS-2 level

**Figure 1** | Parallelization strategy at the MCS-2 level.

run but there is a speedup if multiple simulation runs are required. We define this level as MCS-2 level (**M**odel **C**oordinating **S**oftware at level **2**).

In this investigation, we put emphasis on decreasing the overall runtime by parallelizing programming code on the MCS-2 level. The code base of the MS-1 level is hence left unchanged. This is the typical scenario for a software user who does not have access to the source code or the ability to implement parallel algorithms in the modeling software. A speedup should be achieved by parallel execution of different model simulation runs, regardless of whether the employed modeling software includes parallel or sequential code. Parallel code on both levels is denoted as nested parallelism. This is of special interest in this study as a mutual interference is to be expected for such cases. Figure 1 shows the general parallelization strategy used in this work. The input is an initial model setup (e.g. input-file of SWMM5). The model analysis software performs several analyses by altering the initial setup and simulating the new model setups in parallel (e.g. SWMM5, EPANET2) or nested parallel (e.g. CD3 and par-SWMM).

## Description of case studies

According to the previously defined parallelization strategy, two types of MCS-2 level tools (Achilles and Calimero, see below) are analyzed with regard to their runtime and speedup behavior with increasing number of available cores on a multi core system with shared memory. The speedup is calculated by dividing the measured runtime of the sequential version by the runtime of the parallelized version. While there are many other examples of MSC-2 level tools known and applied, these two are chosen as they cover two distinct features of analysis methods. The most generic MCS-2 level tool would be, for example, a simple Matlab script which sequentially or parallel executes several MS-1 level instances (any model simulation tool).

### Achilles

The first model analysis software and example for a MCS-2 level parallelization is called Achilles (Möderl *et al.* 2010). Achilles is a module for the open source environment SAGA GIS and implemented in C++. The module emulates hazardous events on urban water infrastructure (operational failures, sabotage, land-use change, climate change, earthquakes, debris flows, fluvial flooding, etc.) and performs a spatial sensitivity analysis. According to the type of hazard, defined model adaptations are first performed automatically, then simulated (e.g. sequentially setting increased fire flow demand on each demand junction in a water supply network) and lastly, assessed in terms of hydraulic performance. Each model adaption represents a new scenario, which could be a potential candidate to become representative for a hazardous event with a negative impact on the urban water infrastructure. To analyze the whole system, this step has to be repeated for each infrastructure element (e.g. for each pipe or junction in a water supply network model or for each conduit or node in a combined sewer system model). This results in a huge amount of independent model variations and simulations, which can be carried out in parallel. The number of required model simulation runs is even higher if Achilles analyzes cascading effects of hazardous events within an infrastructure network (Sitzenfrei *et al.* 2011). For this work, Achilles was used to analyze hazardous events on water supply networks (Case study one – CS1) and combined sewer systems (Case study two – CS2) by adapting and simulating EPANET2 and SWMM5 models. The communication between Achilles (MCS-2 level) and the modeling software (MS-1 level) is made true by reading and writing output and input files of the modeling software, respectively.

In CS1.1, component outages within a water supply network (e.g. pipe burst) are simulated and analyzed. The parallelization is only realized on the MSC-2 level (i.e. only in Achilles itself). Each outage of a component is simulated with the help of EPANET2. With global performance indicators taking into account, for example the hydraulic performance, the impact of the component failure is evaluated. All simulations can be performed simultaneously. To point out model size-dependent effects on the speedup, all computational performance tests are based on synthetic water supply networks of different sizes. Therefore, a set of 2,280 water supply models (Möderl *et al.* 2011b) ranging from 26 up to approximately 4,000 nodes were used.

In CS1.2 component failures within a combined sewer system (e.g. conduit collapse) are simulated and analyzed. Here, a combined sewer system that is located in an alpine city with approximately 120,000 inhabitants (model size of 5,400 nodes) is used as an example. The reason for testing the computational performance in such a test setup with only one model (no artificial case studies) is the comparatively high runtime of SWMM5 models. Hydrodynamic storm water simulation generally results in much higher computational effort as compared to EPANET2 simulations, which prevents us here from testing a large set of models. Parallelism in this case study is realized on both levels (nested parallelism) in which each component outage is represented as different SWMM5 input file and simulated with par-SWMM (parallel version of SWMM5) simultaneously. The runtime and speedup tests are realized with increasing the number of used cores of Achilles and par-SWMM (e.g. two cores used by Achilles and three cores used by par-SWMM resulting in six used cores in total).

## Calimero

The second model analysis software and example for a MCS-2 level parallelization is called Calimero (Kleidorfer *et al.* 2009). Calimero can calibrate any model as long as its software is controllable over text-based input and output files or the source code of the model software is available for embedding the model in Calimero. In the second case communication between Calimero (MCS-2 level) and the modeling software (MS-1 level) can be realized without reading and writing files. In this work, communication between Calimero and the modeling software is performed with reading and writing output and input files of the modeling software, respectively. The required number of model alterations and simulations can be independent or dependent on the adopted calibration algorithm. The used optimization algorithm in this work is a parallel particle swarm algorithm (PSO) (Eberhart & Kennedy 1995). Each particle within a swarm, represented by one simulation of the modified initial model setup, is independent of each other particle. In this case, one movement step of the swarm can be calculated in parallel by simultaneously executing several instances of the modeling software. The parallelization strategy is exactly the same as in Achilles.

For this work, the runtime and speedup of a parallel PSO is tested by calibrating EPANET2 (CS2.1), SWMM5 (CS2.2) and CD3 (CS2.3) models alongside a varying number of used threads on a multi core system. In CS2.1 and CS2.2 parallelism is only on the MCS-2 level. In CS2.3 parallelism is applied on both levels and thus serves as an example for nested parallelism. The used models are all from the same region which is a water supply model, a hydrodynamic combined sewer system model and a conceptual combined sewer system model of the alpine city that also is used in case study CS1.2.

## Test environment (TE)

The performance tests (runtime and speedup) of the two case studies were performed on the following test environment: Intel® Xeon® Processor X5650 (TE). The system has two Intel® Xeon® ProcessorX5650 @ 2.67 GHz and 24 GB of DDR3 ram. Each processor has 12 MB L2 cache and six cores running 12 hardware threads in Hyper-threading mode. The installed operating system is Arch Linux using the Linux kernel version 2.6.39-ARCH.

## RESULTS

Runtime and speedup tests of all case studies are presented below along with an interpretation and discussion of their value.

### Achilles (CS1.1)

Figure 2 shows the runtime of Achilles while analyzing component failures of 2,280 EPANET2 models. For all three subfigures the *x*- and *y*-axes represent the number of total threads used by Achilles and the runtime, respectively. Based on the runtime of a sequential execution of Achilles (using one thread only), three categories are defined containing small, medium and large sized models. The median runtime of the small, medium and large sets are approximately 3.4, 14 and 200 s by using one thread. Comparing these values with the related model sizes, it was observed that the small category contains 114 models (26–70 nodes), the medium category contains 342 models (70–210 nodes) and the large category contains 1,824 models (210–4,000 nodes).

Figure 3 shows the speedup of all three categories. The maximum median speedup in the small and large categories

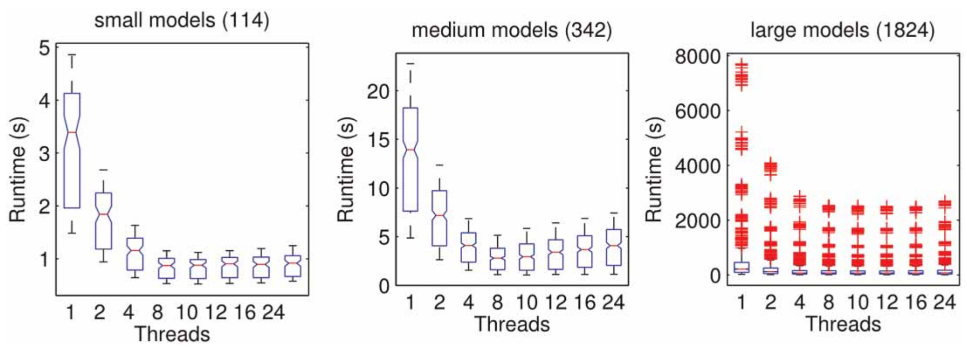

**Figure 2** | Runtime of Achilles on TE testing 2,280 models of the set of synthetic generated water supply networks (CS1.1).

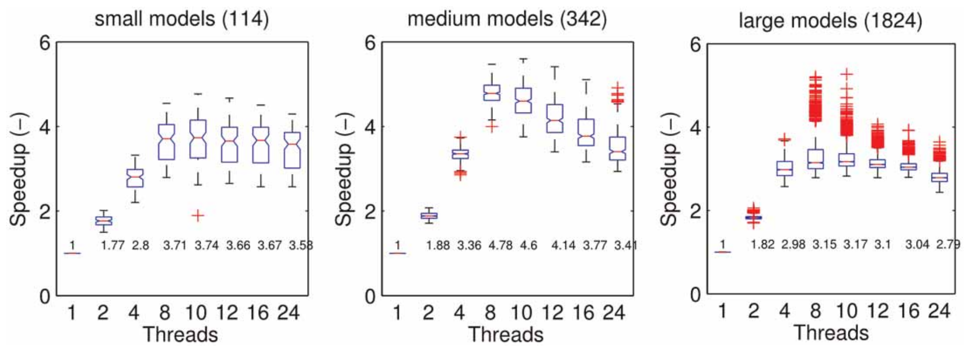

**Figure 3** | Speedup of Achilles on TE testing 2,280 models of the set of synthetic generated water supply networks (CS1.1).

is at 3.74 and 3.17 by using 10 threads. In the medium category the median speedup is even better with 4.78 by using eight threads.

Assuming that the used parallelization strategy is embarrassingly parallel there should be a linear speedup up to 12 used threads (number of real physical cores on the test environment). Between 12 and 24 threads there should also be a speedup increase but not as high as is expected between one and 12 cores because of the hyper threading mode. However, the box plots in Figure 3 show that there must be another boundary for the speedup maximum because the peak is at eight and 10 used threads instead of 12.

$$S(N) = \frac{1}{(1-P) + P/N} \tag{1}$$

This behavior can be explained with the help of Amdahl's law (Equation (1)), which predicts the theoretical maximum speedup $S(N)$ by knowing the number of cores $N$ on a multi core system and $P$ the fraction of parallel code in the program. According to the medium category in Figure 3, the median highest speedup is 4.78 by using eight threads. Upon solving the Amdahl's law equation for the unknown $P$, it is determined that the Achilles program contains approximately 90% of parallel executed code for this example. Figure 4 shows the percentage of parallel executed code of all models by solving the Amdahl's law equation for the unknown $P$ for all three categories. We can see that the percentage of parallel executed code is dependent on the number of used threads and is not constant which means the program contains both a constant overhead and a thread depending overhead.

The problem is that an increase of the EPANET2 model size results in an increase of the fraction of parallel code in Achilles. This subsequently increases the data communication between EPANET2 and Achilles. However, in the current implementation, data communication is sequential if more threads are trying to enter the corresponding code section. Further investigations revealed that the bottleneck is not reading and writing files of EPANET2 result and input files. Instead the bottleneck is a critical code section within Achilles, which converts the results of the model simulation into the internal SAGA GIS data structure. Exactly this conversion is not thread safe and hence cannot be parallelized. The corresponding code can only be entered one thread at a time. Also, with increasing model size the time spent in this code section increases. With increasing number of used threads by Achilles the probability increases that more than one thread is trying to enter the critical code section. This results in a sequential execution of the code and in a decrease of parallel executed percentage of code.

## Achilles with nested parallelism (CS1.2)

In this test case parallelism is applied on the MS-1 and MCS-2 level. The model analysis software applies a SWMM5 model by parallel executing par-SWMM instances. In Figure 5 the $x$-axis in both diagrams show the total number of used threads by Achilles and par-SWMM software. Different numbers of threads used by par-SWMM are indicated with different markers. If par-SWMM uses one thread for each instance, the numbers on the $x$-axes are equal to the numbers of used threads by Achilles. If par-SWMM uses four threads for each instance, the number of threads used by Achilles is the

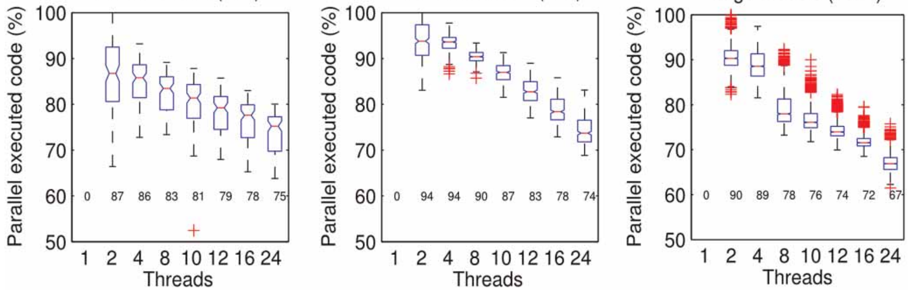

**Figure 4** | Parallel executed code of Achilles by solving the Amdahl's law equation with known speedup and number of used threads (CS1.1).

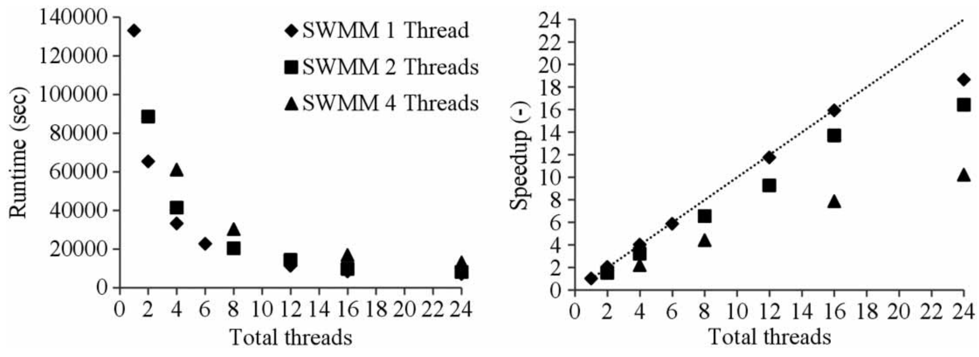

**Figure 5** | Speedup of Achilles analyzing a SWMM5 model by executing par-SWMM instances in parallel on TE (CS1.2) (nested parallelism).

number of total threads divided by four (Figure 5, triangle marker – par-SWMM 4 threads). There is still an increase in speedup when applying nested parallelism (Figure 5 – par-SWMM 4 threads). Compared to the version with parallelism only on the MCS-2 level, the speedup behavior is better (Figure 5, diamond marker – par-SWMM 1 thread). The reason being that the efficiency in terms of processor load of the single threaded par-SWMM version is higher than the multi-threaded version and therefore parallelism only on the MCS-2 level can better utilize all available cores on a multi-core system. Comparing the resulting speedup with CS1.1 we can see that Achilles using par-SWMM with one thread for par-SWMM and 1–24 threads for Achilles is scaling slightly better. The reason for this is because of the smaller critical code section, which converts the SWMM5 results into the Achilles data structure. In this example the runtime spent within MS-1 level is much higher than in CS1.1 and therefore data communication has a minor impact.

## Calimero

Figures 6 and 7 show the runtime and speedup test of Calimero (MCS-2 level) calibrating EPANET2 (CS2.1) and SWMM5 (CS2.2) models, respectively. Since most calibration algorithms contain heuristic mechanisms to determine the set of possible solution candidates, the required number of iterations must vary. Therefore, the runtime is measured by samples (number of iterations or number of tested possible solution candidates) per second. The maximum speedup obtained by calibrating an EPANET2 model with Calimero is approximately eight when using 24 threads (Figure 6). One kink occurs at 12 used threads with a speedup of seven. There is nearly a linear speedup up to 12 threads, which is the number of physical cores on the test environment. By using more threads than available cores (between 12 and 24 threads), speedup once again increases but not that high. This is because of the hyper threading mode in which resources

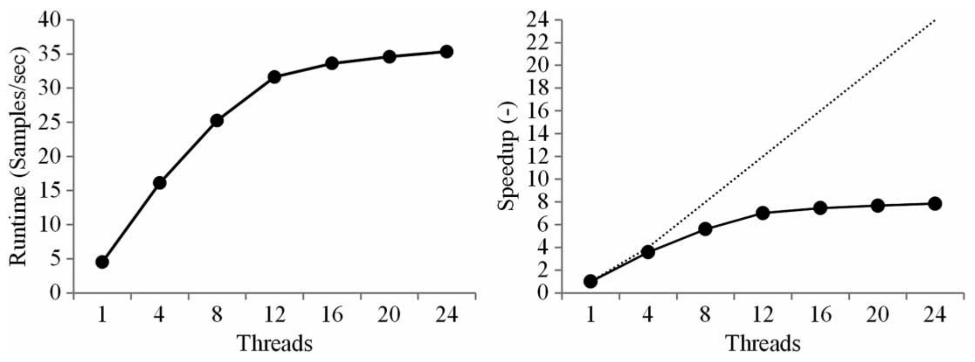

**Figure 6** | Runtime (samples/sec) and speedup of Calimero calibrating an EPANET2 model on TE (CS2.1).

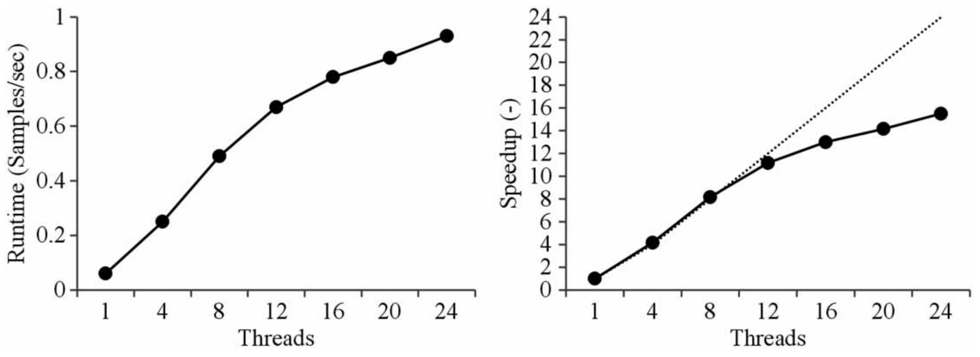

**Figure 7** | Runtime (samples/sec) and speedup of Calimero calibrating a SWMM5 model on TE (CS2.2).

are shared (e.g. floating point unit – FPU). However, results are showing that an increase in speedup is still possible.

The reason for the better performance of CS1.1 compared to CS2.1 is that there is no critical code section within Calimero which is not thread safe. Additionally, only parts of the EPANET2 result files are read. Here, data communication between MCS-2 level and MS-1 level is less time consuming as compared to the Achilles case study. Solving the Amdahl's law equation again shows that the percentage of parallel executed code is nearly constant in the range of real physical cores, which is approximately 92%.

Figure 7 shows runtime and speedup tests for Calimero using SWMM5 models. The layout of the speedup curve is similar to the speedup curve in CS2.1. Again, the highest speedup is at 24 used threads with a factor of 15. A kink at 12 used cores occurs with a speedup value of nearly 12. Up to 12 used threads the speedup is nearly linear. Between 12 and 24 used threads the speedup increases but not that high. We can conclude Calimero calibrating a SWMM5 model is embarrassingly parallel and has nearly 100% of parallel executed code. This is due to the fact that the runtime of SWMM5 models is so high compared to the communication overhead that the latter is of minor importance.

## Calimero with nested parallelism (CS2.3)

In this test case parallelism is applied on the MCS-2 and MS-1 level. The model analysis software calibrates a CD3 model by parallel executing CD3 instances (Figure 8). The *x*-axis in both diagrams shows the total number of used threads by Calimero and CD3. Again, applying nested parallelism does not increase the speedup (Figure 8, triangle marker – CD3 12 threads). Applying parallelism only on the second level utilizes all available cores better and therefore has a better speedup behavior (Figure 8, diamond marker – CD3 1 thread).

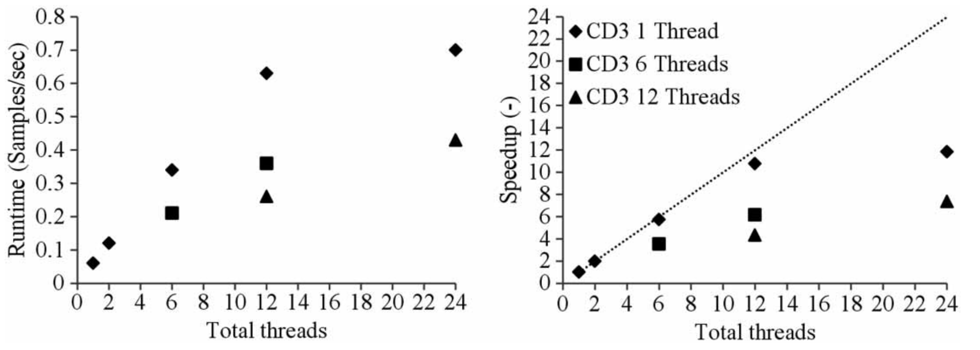

**Figure 8** | Speedup of Calimero calibrating a CD3 model on TE (CS2.3) (nested parallelism).

## CONCLUSION

In this manuscript the potential of a simple parallelization strategy is outlined for analysis software (**M**odel **C**ontrolling **S**oftware at level 2 – MCS-2 level), which inherently needs multiple simulations. The parallelization strategy executes several modeling software instances in parallel (the software itself implemented either sequentially or parallel (**M**odeling **S**oftware at level **1** – MS-1 level). This strategy is shown and analyzed based on runtime and speedup using two different analysis software products denoted as Achilles and Calimero. The investigations show that the speedup of model analysis software with parallelism on the MCS-2 level is increased significantly.

For all case studies data communication between MCS-2 and MS-1 level was realized with reading and writing output and input files of the modeling software. Case studies with Achilles showed that a performance improvement is possible by applying parallelism at the MCS-2 level. Speedup tests were performed on a set of 2,280 different sized water supply systems. Therewith, statistical evaluations and bandwidths of results were obtained. Also size dependent characteristics were analyzed by splitting the data set into three categories. It was shown that Achilles has a non-constant overhead depending on the number of used cores. This is because of a critical code section within Achilles, which can only be executed by one thread at a time. Also, with increasing number of used cores the probability increases that more than one thread tries to enter this code section at a time. This results in a decrease of parallel executed code by increasing the number of used threads. In contrast to that our investigations proved that Calimero performs better. The percentage of parallel executed code is constant in the range of real physical cores used, resulting in a nearly linear speedup.

In cases in which parallelism was applied on both levels (nested parallelism) no performance improvement was seen as compared to the parallel version only on the MCS-2 level. All used modeling software (par-SWMM and CD3) are not fully utilizing all given resources at runtime if they are using more threads. The best performance improvement can be obtained by applying parallelism only on the MCS-2 level, which is an important message for practical applications.

In conclusion, performance improvement of numerical model simulations in the field of urban water management can be obtained by parallelizing model simulation runs regardless of whether the modeling software itself is implemented sequentially or parallel. Even more, best speedup results can be obtained by applying parallelism only on the level of the model analyzing software (MCS-2 level). The more the data communication can be decreased, and especially critical code sections eliminated, the higher speedup values are obtained.

## ACKNOWLEDGEMENTS

This work is part of the DynaVIBe project; funded by the Austrian Science Fund (FWF): P 23250-N24.

## REFERENCES

Burger, G., Fach, S., Kinzel, H. & Rauch, W. 2010 Parallel computing in conceptual sewer simulations. *Water Sci. Technol*. **61** (2), 283–291.

Burger, G. & Rauch, W. 2012 Parallel computing in urban drainage modeling: A parallel version of EPA SWMM. *Proceedings of the Ninth International Conference on Urban Drainage Modelling*, Belgrade, Serbia, 4–6 September 2012. Faculty of Civil Engineering, University of Belgrade, Belgrade.

Crous, P., Van Zyl, J. E. & Roodt, Y. 2012 The potential of graphical processing units to solve hydraulic network equations. *J. Hydroinform*. **14** (3), 603–612.

Dagum, L. & Menon, R. 1998 OpenMP: an industry standard API for shared-memory programming. *Comput. Sci. Eng. IEEE* **5** (1), 46–55.

Eberhart, R. & Kennedy, J. 1995 A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. IEEE, Nagoya Municipal Industrial Research Institute, October 4–6, 1995, pp. 39–43.

Gabriel, E., Fagg, G., Bosilca, G., Angskun, T., Dongarra, J., Squyres, J., Sahay, V., Kambadur, P., Barrett, B. & Lumsdaine, A. 2004 Open MPI: Goals, concept, and design of a next generation MPI implementation. In: *Recent Advances in Parallel Virtual Machine and Message Passing Interface*. Springer, Berlin, Heidelberg, pp. 353–377.

Guidolin, M., Kapelan, Z. & Savic, D. 2013 Using high performance techniques to accelerate demand driven hydraulic solvers. *J. Hydroinform*. **15** (1), 38–54.

Hill, M. D. & Marty, M. R. 2008 Amdahl's law in the multicore era. *Computer* **41** (7), 33–38.

Kleidorfer, M., Leonhardt, G., McCarthy, D. T., Kinzel, H. & Rauch, W. 2009 CALIMERO – A model independent and generalized tool for auto calibration. *Proceedings of the 8th International Conference on Urban Drainage Modelling*, Tokyo, Japan, September, 2009. IWA Publishing, London.

Mair, M., Sitzenfrei, R., Kleidorfer, M., Moderl, M. & Rauch, W. 2012a GIS-based applications of sensitivity analysis for sewer models. *Water Sci. Technol*. **65** (7), 1215–1222.

Mair, M., Sitzenfrei, R., Urich, C., Kleidorfer, M., Möderl, M. & Rauch, W. 2012b Performance improvement with parallel numerical model simulations in the field of urban water management. *Proceedings of the HIC 2012 – 10th International Conference on Hydroinformatics*, Hamburg, Germany, 14–18 July 2012. TuTech Verlag, Hamburg.

Möderl, M., Hellbach, C., Sitzenfrei, R., Mair, M., Lukas, A., Mayr, E., Perfler, R. & Rauch, W. 2011a GIS based applications of sensitivity analysis for water distribution models. *Proceedings of the 2011 World Environmental and Water Resources Congress*, May 22–26, Palm Springs, California. ASCE, American Society of Civil Engineering, Reston.

Möderl, M., Sitzenfrei, R., Fetz, T., Fleischhacker, E. & Rauch, W. 2011b Systematic generation of virtual networks for water supply. *Water Resour. Res.* **47** (2), W02502.

Möderl, M., Sitzenfrei, R. & Rauch, W. 2010 Achilles approach to identify vulnerabilities in urban water infrastructure for operation and emergency management. *Proceedings of the IWA World Water Congress and Exhibition*, Montreal, Canada, 19–24 September 2010. IWA Publishing, London.

Moya, Q. V., Popescu, I., Solomatine, D. & Bociort, L. 2013 Cloud and cluster computing in uncertainty analysis of integrated flood models. *J. Hydroinform*. **15** (1), 55–70.

Olukotun, K., Nayfeh, B. A., Hammond, L., Wilson, K. & Chang, K. 1996 The case for a single-chip multiprocessor. *Proceedings of the ACM Sigplan Notices*, Vol. 31. Association for Computing Machinery, New York, pp. 2–11.

Rossman, L. A. 2010 *Storm Water Management Model User's Manual, Version 5.0*. National Risk Management Research Laboratory, Office of Research and Development, US Environmental Protection Agency, Athens, GA, USA.

Rossman, L. A., Information, C. f. E. R. & Laboratory, N. R. M. R. 2000 *EPANET 2: Users Manual*. Lewis A. Rossman, Water Supply and Water Resources Division, Athens, GA, USA.

Sitzenfrei, R., Mair, M., Möderl, M. & Rauch, W. 2011 Cascade vulnerability for risk analysis of water infrastructure. *Water Sci. Technol*. **64** (9), 1885–1891.

Sutter, H. 2005 The free lunch is over: a fundamental turn toward concurrency in software. *Dr. Dobb's J*. **30** (3), 202–210.

First received 30 October 2012; accepted in revised form 19 April 2013. Available online 30 May 2013