Grid Enabled Metamodeling

Dirk Gorissen, Karel Crombecq, Wouter Hendrickx, Tom Dhaene

Antwerp University, Middelheimlaan 1, 2020 Antwerp, Belgium
{dirk.gorissen,wouter.hendrickx,tom.dhaene}@ua.ac.be, karel.crombecq@student.ua.ac.be

Abstract. The process of simulating and optimizing complex mechanical and electronical systems is a very time consuming and computationally intensive task. As a result, metamodeling techniques are often used for the efficient exploration of the design space, as they reduce the number of simulations needed. However, conventially such metamodels are constructed sequentially, without exploiting inherent parallelism. To tackle this inefficient use of resources we propose a framework where modeler and simulator interact through a distributed environment, (using established grid computing techniques) thus decreasing model generation and simulation turnaround time. This paper provides evidence that such a distributed approach for adaptive sampling and modeling is worthwhile investigating. Research in this new field can lead to even more innovative automated modeling tools for complex simulation systems.

1 Introduction

Computer based simulation has become an integral part of the engineering design process. Rather than building real world prototypes and performing experiments, application scientists can build a computational model and simulate the physical processes at a fraction of the original cost. However, despite the steady growth of computing power, the computational cost to perform these complex, high-fidelity simulations maintains pace. For example, Ford Motor Company reports that one crash simulation on a full passenger car still takes about 36 to 160 hours [1]. Luckily, most of these simulations can be reduced to parallel parameter sweep applications. These consist of several instances of the simulator that are run independently for different input parameters or datasets. Due to the inherent parallelism this can be done in a distributed fashion thus significantly reducing "wall-clock" execution time.

For most realistic problems the high computational cost of simulator codes and the high dimensionality of the design space simply prohibit this direct approach, thus making these codes unusable in engineering design and multidisciplinary design optimization (MDO). Consequently, scientists have turned towards upfront approximation methods to reduce simulation times. The basic approach is to construct a simplified approximation of the computationally expensive simulator (e.g.: aerodynamic drag generated by a particular airfoil shape [2]), which is then used in place of the original code to facilitate MDO, design space exploration, reliability analysis, etc. [1] Since the approximation model acts as surrogate for the original code, it is often referred to as a *surrogate model* or *metamodel*. Examples of such metamodels include Kriging models, Artificial Neural Networks, Support Vector Machines, radial basis function models, polynomial and rational models.

The remainder of this paper is structured as follows: In section 2 we discuss the motivation for constructing parametrized metamodels while section 3 gives an overview of similar research efforts and related projects. Section 4 describes the design and prototype implementation of our framework and section 5 some preliminary performance results. We conclude with a critical evaluation and pointers to future work.

2 Motivation

The reasons for constructing metamodels are twofold: On the one hand metamodels are often used for efficient design space exploration, on the other hand they are used as a cheap surrogate to replace the original simulator. When performing an optimum search, the metamodel guides the search to potentially interesting regions (local minima) [2,3]. Once an adequate solution has been found, the model is discarded. When building a global, scalable model, the metamodel itself forms the object of interest. Here the goal is to construct a parametrized metamodel that can entirely replace the original objective function in the design space of interest. This is useful since the metamodel is much cheaper to evaluate. Once constructed the metamodel is retained and the objective function discarded. In this paper we are primarily concerned with the latter.

However, constructing an accurate metamodels is no trivial undertaking. In some cases it remains questionable if a usable metamodel can be constructed at all. Even if an accurate metamodel is feasible, the process of building it still requires evaluations of the original objective function. Therefore, if the process of constructing a metamodel requires, say, 80 function evaluations and each evaluation takes 10 hours the rate at which the design space can be explored is still relatively low. Nevertheless, the authors argue that this is justifiable since it is a one time, up front investment.

To help tackle this bottleneck we propose a framework that integrates the automated building of metamodels and the adaptive selection of new simulation points (sequential design) with the distributed evaluation of the cost function. This framework will build upon previous work in modeling [4,5,6] and distributed systems [7,8].

3 Related Work

Research efforts and tools that integrate modeling and design space exploration techniques with grid computing techniques can be divided into two categories: those catered towards design optimization and those geared towards the building of standalone scalable metamodels. The first category is by far the most populous. First we have, usually commercial, integrated systems that model and optimize application specific problems. Examples are modeFRONTIER [9] for ship hulls and FlightLab [10] for aircraft.

On the other hand there are many general optimization frameworks which can be applied to different problem domains. The most notable again being Nimrod/O [11]. Nimrod/O is based on the Nimrod/G broker [12] and tackles its biggest disadvantage. This is that Nimrod/G will try to explore the complete design space on a dense grid. This is usually intractable for realistic problems. Nimrod/O performs a guided search through the design space trying to find that combination of parameters that will minimize (maximize) the model output. To this end Nimrod/O employs a number of search algorithms (e.g.: P-BFGS, Simplex, Simulated Annealing). A similar project is DAKOTA [13] from Sandia Laboratories. It is a C++ toolkit that allows a user to choose different optimization algorithms to guide the search for optimal parameter settings. Other projects include GEODISE [14], The Surrogate Management Framework (SMF), SciRun and its precursor Uintah, NetSolve, NEOS and the work by Y. S. Ong et al [15,16].

While all projects mentioned above are tailored towards optimization, they are not concerned with creating a metamodel that can be used on its own. Research efforts that do build replacement metamodels exist [17,18,3,4] but fail to include concepts of distributed computing. Thus the repetitive process of evaluating the objective function while constructing the metamodel is done in a sequential fashion, and this can be extremely time consuming. We were unable to find evidence of other real projects that tackle this. Perhaps the project that comes closest to what we wish to achieve is described in [19], though it too is biased towards optimization.

4 The Design

In this section we outline the architecture of the framework we have designed. A high level design diagram is shown in figure 1. The workflow is as follows: Given an application specific simulator (i.e. the objective function) and a number of model (or algorithm) specific tuning parameters the modeler will build a metamodel with the required accuracy level. In order to do so it interacts with the simulator through the Application Aware Scheduler (AAS) which executes the simulator on the grid through an existing grid middleware or broker.

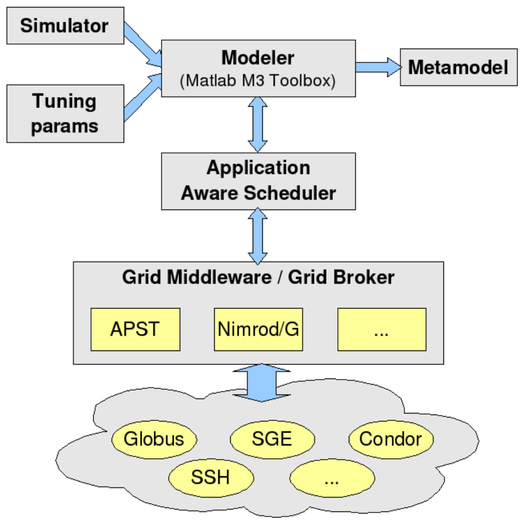

Fig. 1. High level components

4.1 The Modeler

The first component of the framework is the modeler. This component interacts with the simulator in order to construct a meta representation of the original objective function. Our modeler of choice is the Matlab M^3 toolbox [6] and its schematic flowchart is shown in figure 2. The box on the left represents the simulation backend, the component responsible for evaluating the samples. This is the part that will be distributed. The center box depicts the core loop of the toolbox, it performs the adaptive modeling and the selection of new samples. The rightmost box shows the modeler itself. It is responsible for building the polynomial/rational/Neural Network/... metamodels.

The main modeling loop goes as follows: First, an initial sample distribution in the input space is chosen, and simulations are run for all points in this initial sample set. Then, several models of different complexity are built based on the locations of the sample points and the corresponding outputs of the simulator. The models are then compared over a grid, and ranked according to their estimated accuracy. The best models are kept, and a new set of sample locations is adaptively chosen (sequential design). The new sample set is passed on to the simulation backend, which will call the simulator for each of the new sample points. After that, the whole loop repeats itself and will continue until the toolbox has reached the desired accuracy.

For a more detailed treatment of how the modeler works (sample selection, termination criteria, etc) please refer to [5].

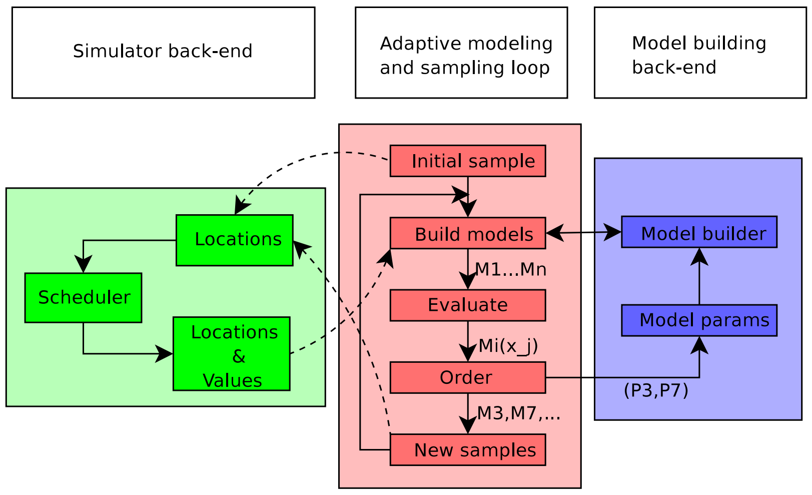

Fig. 2. The **M**ultivariate **M**eta**M**odeling toolbox (M^3)

4.2 The Grid Middleware

In order to distribute the simulation backend across heterogenous resources an extra software layer is needed. For this APST [20,21] is used (though other middlewares such as ProActive, CoBRA, Nimrod/G and Gridbus will be supported in the future). APST can use remote machines accessed through either a Globus GRAM or ssh, remote storage accessed through a Globus GASS server, scp, ftp, sftp, or an SRB server, and queuing systems controlled by Condor, DQS, LoadLeveler, LSF, PBS, or SGE. Since APST grew from AppleS (APPlication LEvel Scheduling) [22] it also tries to schedule jobs optimally based on the characteristics of the resources used. To do this it makes use of established grid information services such as Ganglia, MDS and NWS.

APST consists of an APST client (`apst`) and an APST daemon (`apstd`), both which may be running on separate machines. The client may connect to the server and submit jobs or query the server for job status information. To submit jobs or add resources to the resource pool the client generates an XML file which is then sent to the server.

4.3 Application Aware Scheduler

The Application Aware Scheduler (AAS) is the glue between the modeler and the middleware. It is responsible for translating modeler requests (i.e. evaluations of datapoints) into middleware specific jobs, in this case APST `<task>` tags, polling for results, and returning them to the modeler. The current implementation of the AAS provides a simple Java implemented bridge for mapping between the M^3 toolbox and `apstd`. This already makes it possible to apply our framework to complex realistic modeling problems that were too cumbersome to perform in the sequential case.

As research continues the intelligence of the AAS will be constantly improved. Instead of the simple bridge it is now we will include application specific and resource specific knowledge into the scheduling decision. Rather than requesting a batch of samples to be evaluated with equal priority the modeler assigns scores to each data sample (ie. data samples corresponding to interesting features of the objective function, such as minima and maxima, will receive higher scores). The AAS can then take these priorities into account when making scheduling decisions. Likewise, the AAS should make use of resource information in order to achieve an optimal $task - host$ mapping (ie. complex tasks should be scheduled on the fastest nodes).

5 Performance Comparison

5.1 Experimental Setup

In this section serves to illustrate the application of our prototype framework to an example from Electromagnetics (EM). We will model the the problem twice, once sequentially and once in parallel, and compare the performance. The simulator, for which we shall build a parametrized, scalable metamodel, computes the scattering parameters for a step discontinuity in a rectangular waveguide. The inputs consists of input frequency, the gap height and the gap length. The (complex) outputs are the scattering parameters of this 2-port system. Figure 3 shows an approximate plot of the input-output relation at three different discrete frequencies.

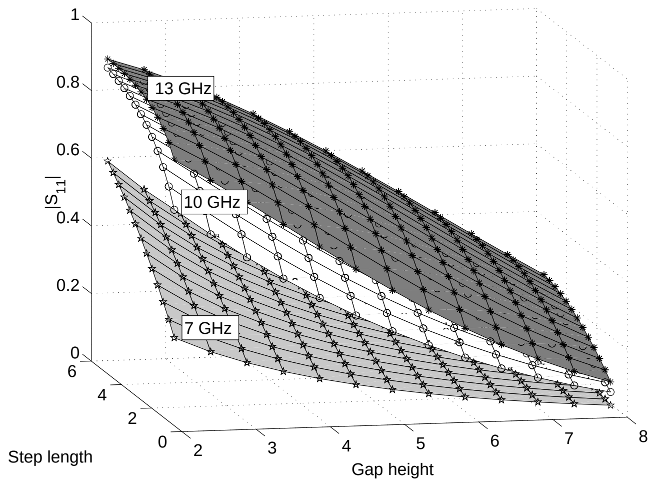

Fig. 3. A Plot of $|S_{11}|$, the Modulus of the First Scattering Parameter

While a real world example, this simulator is still relatively cheap to evaluate. One evaluation takes in the order of 8-13 seconds. Once we are satisfied with the plumbing of the underlying framework we will turn to heavier problems with evaluation times in the order of minutes to hours.

Due to the characteristics of the simulator the exact hardware characteristics of the testbed are of little importance. Nevertheless we list them for completeness. The standalone case was run on a Pentium IV 1.9GHz with 768MB main memory. For the distributed case we used 6 hosts accessible through the local LAN. These included: four Pentium IV 2.4 GHz, one AMD Opteron 1.7 GHz, and one Pentium M 1.6GHz, each with 512MB RAM. While we regret not having been able to use 'real' grid resources we note that this is not really an issue since (1) we are currently only interested in a proof-of-concept (2) we expect the speedup (distributed vs sequential) to increase linearly with the amount of hosts, thus adding more hosts will simply increase the speedup factor.

The M^3 toolbox and `apstd` ran on the same host and the APST scheduling algorithm was the default simple work-queue strategy. No grid information service was configured.

For each case the M^3 toolbox was used to build a metamodel for the objective function described above. We recorded the total time needed to build the model, the time needed to evaluate each batch of samples that is requested as the modeling progresses, and the total number of function evaluations. The results were averaged over three runs.

5.2 Test Results

Figure 5 summaries the different results for each of the runs. If we first look at the average time to process one sample batch we find it is about 56 seconds in the sequential vs 14 in the distributed case. Thus we have an effective speedup factor of about 4. The size of each batch varies between about 3 to 6 samples (due to the non-deterministic nature of the modeler), with the first batch always by far the largest (27).

We notice something similar if we look at the total execution times for each run in figure 4. The total time in the distributed case is about 4 times smaller than in the sequential case for a comparable number of function evaluations.

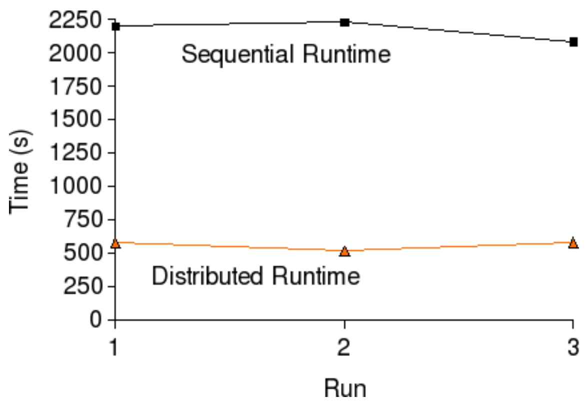

Fig. 4. Comparison Sequential and Distributed

Run	# Samples	Avg Time per Sample Batch (s)	Total Runtime (s)
1	217	56.39	2199.21
2	221	55.75	2229.86
3	206	56.29	2082.66
Avg	**214.67**	**56.14**	**2170.58**

Sequential

1	212	14.91	583.55
2	206	13.56	517.12
3	227	13.82	582.45
Avg	**215**	**14.1**	**561.05**

Distributed

Fig. 5. Test Results: Sequential (top) and Distributed (bottom)

The figure 4 seems illogical since 6 hosts were used in the distribution. One would expect a value of 6 (or in general N if N hosts were used). The reason is that the M^3 toolbox is not yet resource aware (*see* section 4.3) which results in an underutilization of the available compute nodes. With this improvement, together with the move to proper distributed setups involving Globus and SGE administered resources, we expect to improve the speedup significantly in the very near future. Nevertheless these figures still stand since their purpose was to illustrate the integration of distributed computing, adaptive metamodelling and sequential design.

6 Evaluation & Future Work

In this paper we have made the case for the use of gridcomputing techniques while building scalable metamodels. We have presented a prototype framework based on the M^3 toolbox and APST and contrasted its performance with the traditional approach of analyzing datapoints sequentially. The initial results look very promising and warrant further extension of our framework in the future.

Future work will include:

- Move to real distributed setups involving Globus and SGE administered clusters.
- Creation of a 'real', pluggable framework where the user will be able to easily choose the modeling algorithm and the grid middleware to suit his or her application. In addition, if a required capability is not available the user should be able to plug in his own extension.
- Apply AI techniques such as genetic algorithms and machine learning algorithms to enhance the modeling, decrease the reliance on simple heuristics, and allow for more automated tuning of the modeling process.

References

1. Simpson, T.W., Booker, A.J., Ghosh, D., Giunta, A.A., Koch, P.N., Yang, R.J.: Approximation methods in multidisciplinary analysis and optimization: A panel discussion. Structural and Multidisciplinary Optimization **27**(5) (2004) 302–313
2. Marsden, A.L., Wang, M., Dennis, J.J.E., Moin, P.: Optimal aeroacoustic shape design using the surrogate management framework: Surrogate optimization. Optimization and Engineering **Volume 5**(2) (2004) pp. 235–262(28)
3. Martin, J.D., Simpson, T.W.: Use of adaptive metamodeling for design optimization. In: Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA. (2002)
4. Hendrickx, W., Dhaene, T.: Multivariate modelling of complex simulation-based systems. Proceedings of the IEEE NDS 2005 Conference (2005) 212–216
5. Hendrickx, W., Dhaene, T.: Sequential design and rational metamodelling. In Kuhl, M.E., M., S.N., Armstrong, F.B., Joines, J.A., eds.: Proceedings of the 2005 Winter Simulation Conference. (2005) Accepted.
6. Hendrickx, W., Dhaene, T.: m^3-toolbox (2005) Available on `www.coms.ua.ac.be` in the `Software` section.

7. Gorissen, D., Stuer, G., Vanmechelen, K., Broeckhove, J.: H2O Metacomputing - Jini Lookup and Discovery. In: Proceedings of the International Conference on Computational Science (ICCS), Atlanta, USA. (2005) 1072–1079
8. Hellinckx, P., Vanmechelen, K., Stuer, G., Arickx, F., J., B.: User experiences with nuclear physics calculations on H2O and on the BEgrid. In: in Proceedings of the International Conference on Computational Science (ICCS), Atlanta, USA. (2005) 1081–1088
9. modeFRONTIER: (modeFRONTIER homepage) http://www.esteco.it/Products/.
10. FlightLab: (Flightlab homepage) http://www.flightlab.com/.
11. Abramson, D., Lewis, A., Peachey, T., Fletcher, C.: An automatic design optimization tool and its application to computational fluid dynamics. In: Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM), New York, NY, USA, ACM Press (2001) 25–25
12. Abramson, D., Giddy, J., Kotler, L.: High performance parametric modeling with Nimrod/G: Killer application for the global grid? In: Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS), Cancun, Mexico. (2000) 520– 528
13. Giunta, A., Eldred, M.: Implementation of a trust region model management strategy in the DAKOTA optimization toolkit. In: Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA. (2000)
14. Eres, M.H., Pound, G.E., Jiao, Z., Wason, J.L., Xu, F., Keane, A.J., Cox, S.J.: Implementation and utilisation of a grid-enabled problem solving environment in matlab. Future Generation Comp. Syst. **21**(6) (2005) 920–929
15. Ng, H.K., Lim, D., Ong, Y.S., Lee, B.S., Freund, L., Parvez, S., Sendhoff, B.: A multi-cluster grid enabled evolution framework for aerodynamic airfoil design optimization. In: ICNC (2). (2005) 1112–1121
16. Ng, H.K., Ong, Y.S., Hung, T., Lee, B.S.: Grid enabled optimization. In: EGC. (2005) 296–304
17. Lehmensiek, R., Meyer, P.: Creating accurate multivariate rational interpolation models for microwave circuits by using efficient adaptive sampling to minimize the number of computational electromagnetic analyses. IEEE Trans. Microwave Theory Tech. **49**(8) (2001) 1419–
18. De Geest, J., Dhaene, T., Faché, N., De Zutter, D.: Adaptive cad-model building algorithm for general planar microwave structures. IEEE Transactions On Microwave Theory and Techniques **47**(9) (1999) 1801–1809
19. Parmee, I., Abraham, J., Shackelford, M., Rana, O.F., Shaikhali, A.: Towards autonomous evolutionary design systems via grid-based technologies. In: Proceedings of ASCE Computing in Civil Engineering, Cancun, Mexico. (2005)
20. Casanova, H., Obertelli, G., Berman, F., Wolski, R.: The AppLeS parameter sweep template: User-level middleware for the grid. In: Proceedings of Supercomputing (SC 2000). (2000)
21. Casanova, H., Legrand, A., Zagorodnov, D., Berman, F.: Heuristics for scheduling parameter sweep applications in grid environments. In: Proc. 9th Heterogeneous Computing Workshop (HCW), Cancun, Mexico (2000) 349–363
22. Berman, F., Wolski, R., Casanova, H., Cirne, W., Dail, H., Faerman, M., Figueira, S., Hayes, J., Obertelli, G., Schopf, J., Shao, G., Smallen, S., Spring, N., Su, A., Zagorodnov, D.: Adaptive computing on the grid using AppLeS. IEEE Transactions on Parallel and Distributed Systems (TPDS) **14**(4) (2003) 369–382