Parallel Evolutionary Optimization under Matlab on standard computing networks

Hartmut Pohlheim
DaimlerChrysler AG, Research and Technology
Alt-Moabit 96a, 10559 Berlin, Germany
hartmut.pohlheim@daimlerchrysler.com

Sven Pawletta, Andreas Westphal
University of Rostock, Institut for Automatic Control
Richard-Wagner-Str. 31, 18051 Rostock, Germany
sven.pawletta@etechnik.uni-rostock.de

1 INTRODUCTION

For many computing intensive tasks at DaimlerChrysler Research MATLAB [1] is used. Matlab has become a defacto standard in research and development. Many areas are catered for by a wide range of toolboxes and the Simulink non-linear simulation package along with extensive visualization and analysis tools. In addition, Matlab has an open and extensible architecture allowing individual users to develop further routines for their own applications. These qualities provide a uniform and familiar environment on which to use and build extended tools.

Since 1995 we use Evolutionary Algorithms for many difficult optimization tasks. We employ the Genetic and Evolutionary Algorithm Toolbox for use with Matlab (GEATbx) [6]. This toolbox provides a large number of different operators and functions for a wide range of evolutionary algorithms.

For most evolutionary optimization tasks the execution times were satisfying on standard PC's or workstations (some minutes to 1-2 hours). However, when optimizing large systems one run could take more than a day. Such a long optimization time is not acceptable. From experiments with test systems we knew, that the evolutionary part of an optimization run takes only 2-10 minutes. The remaining time is spent on executing the objective function.

Evolutionary Algorithms are inherent parallel. At each generation the same objective function is executed with multiple different data sets (the individuals). This "single program - multiple data" scheme opens the way for shorter optimization times by distributing the calculation of the objective function to multiple machines. Multi processor machines and clusters of networked computers are readily available today, especially networks of standard PC's and UNIX/LINUX workstations.

In 1997 we looked for a tool providing the functionality of distributing computing tasks to multiple machines or processors. Our aim was to include this functionality into Matlab in a transparent way. We appreciate using our familiar and powerful tools, but combined with a comfortable way of distributing calculations to multiple machines.

2 TOOLS AND THEIR INTEGRATION

2.1 DP TOOLBOX

The Distributed and Parallel Processing Toolbox for use with Matlab (DP Toolbox) [4] provided the features we were looking for at this time. (MultiMatlab [2] seemed equally promising, but wasn't available at this time.)

The DP Toolbox for Matlab is one realization of the general "Multi-Instance Concept" described in [3]. This approach brings together the convenience of Matlab and similar systems (especially the interactive way of working) with the power of parallel and distributed processing.

At the lowest level the DP Toolbox extends Matlab by an interaction module. This module provides facilities to control Matlab instances and other programs as well as to communicate between them. The current versions of the DP Toolbox are based on primitives for process control and communication provided by PVM (Parallel Virtual Machine [5]). PVM is an open system and available for a large number of operating systems and hardware platforms including SUN Solaris and WindowsNT.

The DP Toolbox contains several levels of abstraction. The DP low level implements an interface to the underlying PVM system. But for convenient usage in Matlab applications the abstraction degree of the PVM routines is much too low, because they are designed to meet the requirements of classical C or Fortran programming. For sophisticated Matlab applications the DP high level interface is quite more suitable. This interface provides a small number of specialized functions for sending and receiving Matlab data objects (array-passing) and for launching and terminating Matlab instances. Thus, the user of the high level interface needs not to worry about all the implications of the PVM interface.

Pohlheim, H., Pawletta, S. and Westphal, A.: Parallel Evolutionary Optimization under Matlab on standard computing networks. in *Banzhaf, W. (eds.)*: GECCO'99 - Proceedings of the Genetic and Evolutionary Computation Conference - Workshop program, San Francisco, CA: Morgan Kaufmann, pp. 174-176, 1999.

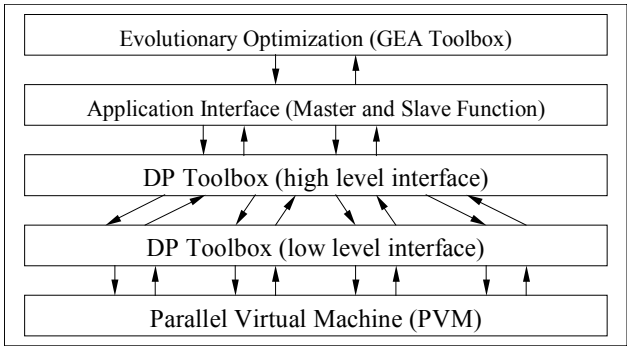

Figure 1: Scheme of the integration and interface between PVM, DP Toolbox and GEA Toolbox

2.2 GEA TOOLBOX

The GEA Toolbox [6] is a comprehensive implementation of Evolutionary Algorithms in Matlab. A broad range of operators is fully integrated into one environment constituting a powerful optimization tool applicable to a wide range of problems. A broader description of the features of the GEA Toolbox is not the intention of this paper. For more information about the GEA Toolbox please consult [6] and [7] or contact the first author.

2.3 INTEGRATION OF TOOLS

For a full integration of the functionality of the DP Toolbox into the environment of the GEA Toolbox (and other single program multiple data tasks under Matlab) an additional application interface was developed. This interface hides even the last details of the data sending and calculation distribution tasks.

Thus, the user does not call the problem specific objective function with multiple data sets. Instead, the master function is called with appropriate parameters (name of objective function, data sets and any additional problem specific parameters). This function starts the virtual machine and the slave processes and distributes the tasks to the configured machines (calling the problem specific objective function with one or multiple data sets). The master function also collects the results and sends them back to the calling function. At the end the slaves and the virtual machine can be closed. Additionally, an efficient load balancing scheme is implemented inside the master function. Thus, the configured machines are used as efficient as possible and slow machines don't slow down the overall calculation time.

All this integration is done by a multi level approach, see figure 1. Each level abstracts parts of the lower level. At the moment only the necessary functionality is implemented. Because of the modular and multi level structure the implementation of additional functions is easy and straightforward.

The initial setup of PVM and the remote access service is currently not totally simple, especially under WindowsNT. However, these processes are continually improved and are much easier now than one year ago.

All integrated tools are under continual development. This includes new features and a higher robustness. We have very good experience with the described setup. The implementation runs satisfactory. We use it in our everyday work and we could solve a number of optimization tasks not manageable before.

3 EXAMPLES AND RESULTS

We used the described Evolutionary optimization setup for the solution of a number of large and complex systems. A prominent example was the parameter optimization of a combustion model of a diesel engine, [8]. The simulation of the combustion model in one scenario with one parameter set took around 2 seconds (depending on the used computer). 14 different scenarios were defined.

Typical optimizations were run over 3-10 scenarios. A 3-scenario optimization with 90 individuals over 100 generations took around 800 minutes or 13.5 hours on one computer. When extending the optimization to 10 scenarios and 150 individuals over 1000 generations the run took 4500 minutes or more than 3 days.

By using 5 machines to optimize this system we could bring down the total computation times to less than 3 hours or 15 hours respectively.

Figure 2 illustrates how the total computation time comes down when the number of used machines gets larger. 100 objective function calls are equivalent to the calculation of one generation. However, when using more than 5 machines, the saved total computation time gets smaller. Additionally, we had only 5 equally quick computers, the additional 5 machines were slow compared to the first 5 machines. Nevertheless, when each generation contains at least 100 individuals, the total computation time can still be shortened.

Figure 3 shows the distribution of the objective function calls to 10 different machines. The contribution of the slower machines is rather small. Nevertheless, as more objective function calls per generation must be computed, even some slow machines can calculate some of the jobs. However, as smaller the number of objective function calls is as equally fast the used machines should be. Otherwise, more machines don't produce a smaller overall computation time.

Pohlheim, H., Pawletta, S. and Westphal, A.: Parallel Evolutionary Optimization under Matlab on standard computing networks. in *Banzhaf, W. (eds.)*: GECCO'99 - Proceedings of the Genetic and Evolutionary Computation Conference - Workshop program, San Francisco, CA: Morgan Kaufmann, pp. 174-176, 1999.

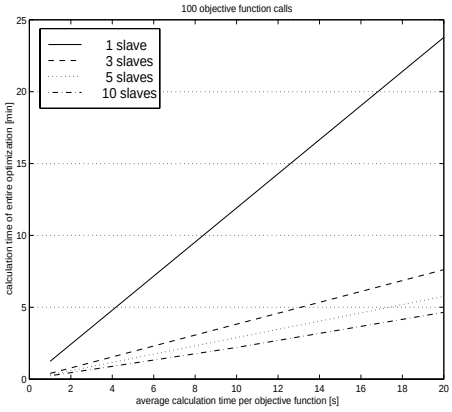

Fig 2: Total computation time for 1, 3, 5 and 10 machines for different calculation times per objective function call (1-20 seconds)

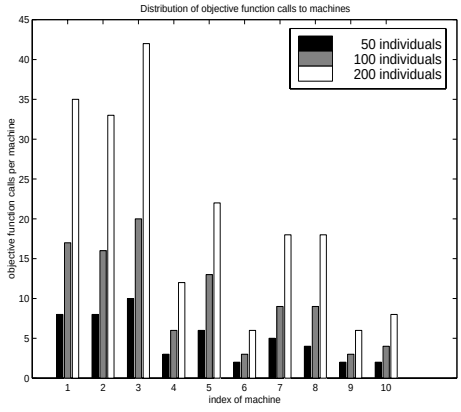

Fig 3: Distribution of objective function calls to 10 different machines for 50, 100 and 200 objective function calls

Other systems we optimized using parallel evolutionary algorithms:

- Optimization of the control of a chopper, [9],
- Optimization of the control strategy for a complex greenhouse system, [10],
- Parameter optimization of different time consuming Simulink models.

4 CONCLUDING REMARKS

The combination of Matlab, the GEA Toolbox, the DP Toolbox, and PVM provides a powerful tool for the distribution of time intensive evolutionary experiments to multiple machines. Heterogeneous operating systems and machine architectures could be used at the same time and in the same computing task. Thus, the available computer infra structure could be easily employed to solve the tasks at hand. No special investment in hardware or expensive software is necessary.

The integration of the DP Toolbox into Matlab and the GEA Toolbox provides a fully transparent interface to the user. Depending on the size of the optimization task the user may use serial or parallel calculation of the objective function. The switch is just one parameter away. No additional special software or setup is necessary. The same tools used every day for evolutionary optimization can be used for parallel optimization.

By employing a different number of machines the time needed for the optimization task can be adjusted. As the results shown above indicate, long optimization tasks could be finished in 3-6 hours instead of more than one day. This brings the comfort back to try different settings. On the other hand we have now the tools to solve evolutionary optimization tasks which were earlier too time consuming. Thus, new challenges could be approached.

References

[1] *MathWorks, The*: Matlab - User Guide. Natick, Mass.: The MathWorks, Inc., 1994-1997. http://www.mathworks.com/

[2] *Trefethen, A. E.*: MultiMatlab. Cornell Theory Center, 1996. http://www.tc.cornell.edu/~anne/projects/MM.html

[3] *Pawletta, S.*: Erweiterung eines wissenschaftlich-technischen Berechnungs- und Visualisierungssystems zu einer Entwicklungsumgebung für parallele Applikationen. Dissertation, Universität Rostock, 1998. (Extension of a Scientific and Technical Computing and Visualization System to a Development Environment for Parallel Applications.)

[4] *Pawletta, S., Westphal, A., Pawletta, T., Drewelow, W., Duenow, P.*: Distributed and Parallel Application Toolbox (DP Toolbox) for use with Matlab - User's Guide and Reference Manual Version 1.4. Institute of Automatic Control, University of Rostock, 1999. http://www-at.e-technik.uni-rostock.de/dp/

[5] Parallel Virtual Machine. Oak Ridge National Laboratory, Computer Science & Mathematics Division. http://www.epm.ornl.gov/pvm/pvm_home.html

[6] *Pohlheim, H.*: Genetic and Evolutionary Algorithm Toolbox for Matlab. http://www.geatbx.com/, 1994-1999.

[7] *Pohlheim, H.*: Entwicklung und systemtechnische Anwendung Evolutionärer Algorithmen. Aachen, Germany: Shaker Verlag, 1998. (Development and Engineering Application of Evolutionary Algorithms. in german)

[8] *Pohlheim, H. und Schütte, A.*: Optimierung der Parameter in einem Verbrennungsmodell für einen Dieselmotor mit Evolutionären Algorithmen. Technischer Bericht FT3/A-1998-001, Daimler Benz AG, 1998. (Parameter optimization of a combustion model of a diesel engine using evolutionary algorithms)

[9] *Pohlheim, H. und Schütte, A.*: Optimierung der Regelung eines Gleichstromstellers mit Evolutionären Algorithmen. interner Technischer Bericht F3S-97-009, Daimler Benz AG, 1997. (Optimization of the control of a chopper using evolutionary algorithms)

[10] *Pohlheim, H. and Heißner, A.*: Optimal Control of Greenhouse Climate using real-world weather data and Evolutionary Algorithms. in *Banzhaf, W. (eds.)*: GECCO'99 - Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA: Morgan Kaufmann, 1999.

Pohlheim, H., Pawletta, S. and Westphal, A.: Parallel Evolutionary Optimization under Matlab on standard computing networks. in *Banzhaf, W. (eds.)*: GECCO'99 - Proceedings of the Genetic and Evolutionary Computation Conference - Workshop program, San Francisco, CA: Morgan Kaufmann, pp. 174-176, 1999.