Component Object Based Single System Image Middleware for Metacomputer Implementation of Genetic Programming on Clusters

Ivan Tanev[1], Takashi Uozumi[1], and Dauren Akhmetov[2]

[1] Department of Computer Science and System Engineering, Muroran Institute of Technology, Mizumoto 27-1, Muroran 050-8585, Japan
i.tanev@computer.org, uozumi@csse@muroran-it.ac.jp

[2] Solution Department, Mobile Multimedia Business Headquarters, NTT DoCoMo Hokkaido, Inc, 6-Nishi-1, West-14, Chuo-ku, Sapporo 060-0001, Japan
dauren_akhmetov@nttdocomo-h.co.jp

Abstract. We present a distributed component-object model (DCOM) based single system image middleware (SSIM) for metacomputer implementation of genetic programming (MIGP). MIGP is aimed to significantly improve the computational performance of genetic programming (GP) exploiting the inherent parallelism in GP among the evaluation of individuals. It runs on cost-effective clusters of commodity, non-dedicated, heterogeneous workstations. Developed SSIM represents these workstations as a unified virtual resource and addresses the issues of locating and allocating the physical resources, communicating between the entities of MIGP, scheduling and load balance. Adopting DCOM as a communicating paradigm offers the benefits of software platform- and network protocol neutrality of proposed implementation; and the generic support for the issues of locating, allocating and security of the distributed entities of MIGP. Presented results of experimentally obtained speedup characteristics show close to linear speedup of MIGP for solving the time series identification problem on cluster of 10 W2K workstations.

1 Introduction

Genetic programming (GP) [1] is an algorithmic paradigm, inspired by the natural evolution of species. GP is successfully applied for solving increasingly difficult problems in artificial intelligence such as electrical circuits design, evolving digital hardware, spatial information classification, time-series identification, etc. [1]. However, GP is computationally costly - running even moderately sized problems, to which GP is applied, often requires hours on currently available computer architectures. One of the ways of speeding-up the GP is to improve the computational performance of the implementation. One of the most promising approaches for improving the computational performance is to exploit the inherent parallelism of GP by mapping the GP on parallel multiprocessor systems. Networks (clusters) of workstations (NOW) as a parallel environment feature most attractive specific cost characteristics

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 284–293, 2001.
© Springer-Verlag Berlin Heidelberg 2001

due to the well-known fact that the prices of the components are based on the manufactured volume. Typically, NOW can be built of commodity "off-the-shelf" components, significantly reducing both the developing time and the costs [2]. Our *objective* is to develop MIGP featuring improved computational performance of GP by exploiting the parallelism among the evaluations of individuals. MIGP must be cost-effective in that to allow for deployment on clusters of commodity, non-dedicated workstations.

Although some work on implementing GP exploiting the parallelism among evaluations of individuals on NOW had been previously done [3], [4], [5], it assumes deployment of developed systems on networks of dedicated, homogeneous workstations. Therefore, the issues of resources location and allocation, scheduling, load balancing, and even more, the issue of creating a single image of the underlying distributed system had not been considered as relevant in these implementations. On the other hand, employing the currently available general-purpose full-featured metacomputer systems [6], [7], [8], [9] for addressing the concrete issue of parallel implementation of GP is viewed by us as too excessive approach. Moreover, although these systems do offer flexibility to the end-users, some concerns could be raised about their ability to be finely tuned to adequately address the communication-intensive nature of the considered model of parallelism in GP. In addition, from viewpoint of the communicating paradigm, the currently available metacomputers are heavily relying on the de-facto-standard MPI/PVM-based communications, while, in our view, the need for investigating the feasibility of recently emerged component object models for engagement as a communication paradigm in clusters seems to be underestimated by the HPC. All these concerns represent the rationale behind our approach to develop and evaluate the component object based SSIM for MIGP.

Addressing the challenge of potentially heterogeneous nature of resources in NOW, we propose a single system image middleware (SSIM), which represents the NOW as a single metacomputer (virtual supercomputer). Distributed component object model (DCOM), adopted as an underlying communicating paradigm contributes to address the heterogeneity of software platforms of workstations and the heterogeneity of network protocols. The resource management and scheduling subsystem of SSIM addresses the issue of efficient utilization of workstations that feature different speed of network connections, computational power, and/or dynamically changeable computational workloads. The reminder of the paper is organized as follows. Section 2 outlines the considered time series identification problem (TSIP), and the GP as an algorithmic paradigm, adopted for solving it. Section 3 discusses the developing of SSIM for MIGP. Also, it enumerates both the benefits and the challenges of adopting DCOM as a communicating paradigm. Section 4 presents performance evaluation results. The conclusion is drawn in Section 5.

2 The Problem and the Algorithmic Paradigm

Identification of a mathematical model of a real process is the necessary step in a wide class of modern control, information processing, diagnosis and related problem solving [10]. Let consider an identification problem statement. Assume that some plant is

isolated from the environment on the grounds of fixed features (Fig. 1). The input *u(n)* and output *y(n)* signals of the plant are supposed measurable at the instants n=1,2,… during functioning. The plant is interfered with random non-observable disturbance *(n)*. There is an unknown operator of input-output mapping of the form:

$$y(n) = A[u(n)] \quad (\mathbf{1})$$

The operator (1) is characterized with unknown structure *St* and parameter vector $\tilde{a}$:

$$A =< St, \tilde{a} > \quad (\mathbf{2})$$

The identification problem is determination of the plant's operator estimate $\hat{A}$ by means of input and output signals measuring and processing. Typically, the criteria to evaluate the quality of the model are based on error between plant's output and model's output. The identification is an optimization problem with an objective to find the minimum of the error between the output of the model and the real output. We consider the identification of the model of vibration data from the crystal-polishing machine as a real-world TSIP. Example of the sample data is shown in Fig. 2.

Requirements of high quality automatic control had made it necessary to use novel approaches for real process description which are often characterized with high dimensionality, substantial non-linearity and unmodelled dynamics. Such class of problems may be solved by soft computing approaches - artificial neural networks, fuzzy systems, chaos computing, immune network computing, genetic algorithms, GP, etc. As the results of many researchers have shown, effective systems may be designed on the basis of fusion of different soft computing approaches [11]. GP (Fig.3), considered in our approach, is an algorithmic paradigm, inspired by the natural evolution of species based on the survival of the fittest [1].

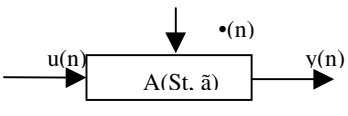

Fig. 1. Operator presentation of a plant

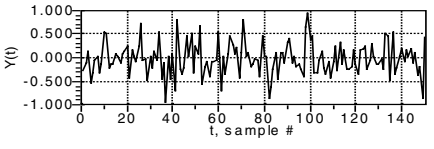

Fig. 2. Vibration data of crystal-polishing machine

```
1: Create the initial population of GPs;
2: Evaluate initial population;
3: while not success predicate do Steps 4..5
4:    Perform reproduction: mating pool creation;
5:    while initial population size not reached do Steps 6..9
6:       Select a pair of GPs from mating pool;
7:       Perform crossover and produce 2 GPs - offspring;
8:       Evaluate offspring;
9:       Incorporate offspring into the population;
```

Fig. 3. Algorithm of GP

GP considers a population of individuals (genetic programs, GPs) that evolves. Each of GPs represents a solution to the problem. The initially (randomly) created population evolves through many generations, applying the main genetic operations - reproduction, selection and crossover, at each generation until the best individual solves the problem with desired precision. The power of GP is in that computational effort of such evolution is much less than in the random search-based approaches. The main parameters of GP for TSIP are shown in Table 1.

3 SSIM for MIGP

3.1 Model of Parallel Implementation of GP

Partitioning of GP. Partitioning considers the way GP is decomposed into smaller, relatively autonomous processes that might be simultaneously executed on multiple workstations. Taking into consideration the algorithm of evolving single generation in GP and the runtime breakdown results obtained from the developed prototype of sequential GP for solving the TSIP, we considered the implementation of parallelism among the evaluations of individuals (Fig. 3, Step 8). The key argument in favor of such a decision is that for the considered case of TSIP the evaluation of individuals consumes more than 98% of GP-runtime.

Communicating Paradigm: DCOM. Exploiting the considered parallelism in GP assumes that the code of evaluation of individuals must be running on workstations, remote with respect to the workstation that manages the GP-population and performs the main genetic operations. Such a code must be implemented as a remotely controllable process, and several communication technologies can be exploited for such a purpose. These include component-object-based (COB) technologies, such as DCOM [12] and CORBA, and non-COB, such as sockets level programming, MPI, PVM, RPC, and Java RMI. Our choice for using COB paradigm is based on the fact that it provides the domain of distributed computing (DC) with many of the same benefits, such as encapsulation, reuse, portability, and extensibility, as it does for non-DC. Furthermore, applied to the domain of DC, these benefits can be extended with features as binary standardization, platform-, machine- and protocol-neutrality, and ability to seamlessly integrate with different Internet protocols. All these features are incorporated into DCOM, which in addition, as a system model offers generic support for the issues of naming, locating and protecting the communicating entities. Adopting DCOM tends to compromise (to certain degree) the communication characteristics of the implementation compared to the widely adopted MPI and PVM. The throughput of DCOM, which we obtained for two protocol stacks – TCP/IP (W2K) and UDP/IP (NT4.0), compared with the throughput of several implementations of MPI on NT4.0 [13] is shown in Fig. 4. The CPU for all of the considered cases is 450MHz Pentium II, and the underlying network is 100 Mbps Fast Ethernet. As figure illustrates, for small messages DCOM is more than 2 times slower than NT-MPICH – the fastest

implementation of MPI. For larger messages however, the throughput of DCOM on W2K (over TCP/IP) is only about 20% off the throughput of NT-MPICH.

Table 1. GP for TSIP

Objective	For given data source $Y(t_i)$, i=1,2,..2000 to find the analytical expression $Y(t_i)=F(S_T)$ which fits the given data source with specified error. S_T is the terminal set.
Terminal set	S_T =[$Y(t_{i-1})$, $Y(t_{i-2})$,…$Y(t_{i-P})$, C, A, M], where C is a random constant from the interval [0,1], M and A are the average and absolute average amplitude respectively. P=10.
Function set	[+, -, *, /]
Fitness cases	The given sample of 2000 data points.
Raw fitness	The average, taken over 2000 fitness cases, of quadratic value of the difference between value of the $Y(t_i)$, produced by individual (S-expression) and the target value $Y(t_i)$ of the given data source.
Reproduction ratio	20% - reproduction, 80% - selection (both – fitness-proportional)
Success predicate	Raw fitness is less than 0.01
Population size	1000 individuals

Challenges in Adopting DCOM. Batching. Implementing the proposed parallelism in GP exploiting DCOM implies that the evaluation of individuals is performed by remote DCOM-server, which runs on a separate workstation. The routine of evaluation of individuals is remotely invoked by the client (GP manager - GPM), which runs on separate client machine. Client submits the GPs to DCOM-server by invoking the corresponding methods of DCOM-server. The invocation is location transparent, platform-, and protocol-neutral, which allows for deployment of our implementation on network of heterogeneous workstations. However, the cost of these benefits is that, performance-wise compared to MPI and PVM, such a remote method invocation suffers from significant software overhead, and less data transmission rate (refer to Fig.4). For example, the typical software overhead of DCOM is in order of several hundreds •s, which is comparable with the computational cost of evaluating simple individuals.

We applied batching (i.e. bundling multiple method invocations into a single one) both in submission of the individuals and in forwarding the fitness values back to the client. The experimentally obtained effect of batching on the latency of submission of individuals with complexity of 100 tree nodes, which is close to the average for the GP-populations over many independent runs is shown in Fig. 5. As figure illustrates, with increasing of the batch size, the specific latency (*Ls*) of submitting single individual within the batch decreases, and for batch size of 40 individuals is about 8 times less than latency of submitting single individual *L(1GP)*. However, we consider the value of 16 individuals as an optimal batch size for our implementation. As depicted in Fig. 5, larger batches result only marginal improvement of the specific latency. In addition, larger batches imply increasing the granularity of scheduled tasks, which might increase the degree of unbalancing of the workload when the total amount of

batches cannot be evenly distributed among the available workstations. The effect of batching on the load balancing is considered later in following subsection 3.2.

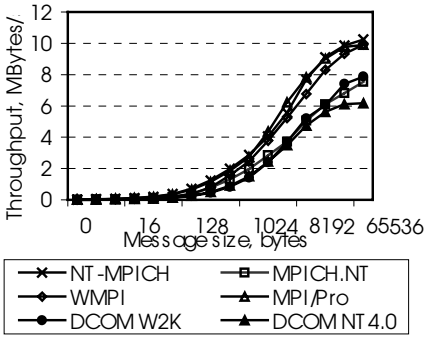

Fig. 4. Throughput of DCOM compared with various implementations of MPI on NT4.0

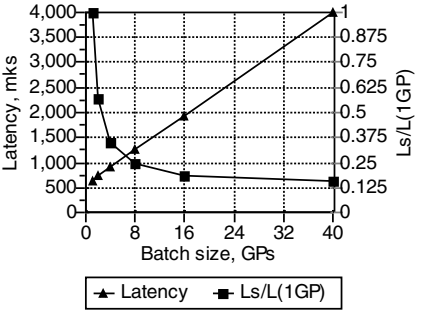

Fig. 5. Effect of batching on the latency of submitting GP-individuals

3.2 Distributed Architecture

Exploiting the considered model of parallelism implies that the routine of evaluation of the individuals is performed by Evaluation Servers (ES), running on separate workstations in the cluster. An eventual straightforward two-tiered implementation, where the client (GPM) besides the main genetic operations handles the resource management and scheduling (RMS) would raise modularity-, flexibility- and manageability related concerns. To address these concerns, we propose a three-tiered architecture incorporating the single system image middleware (SSIM) with functionality solely devoted to the issues of RMS. The architecture of MIGP is shown in Fig. 6. The functionality of the entities of the proposed architecture is as follows.

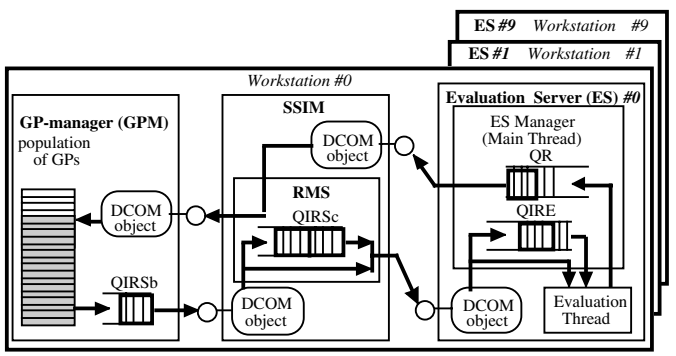

Fig. 6. Distributed architecture of MIGP

GPM. Manages the GP-population, and performs genetic operations – reproduction, selection, and crossover. Using a queue of individuals, ready to be submitted (QIRSb), packets the pairs of offspring into a single batch. Upon completing QIRSb submits the batch to SSIM. Incorporates the results of evaluation upon receiving them from SSIM.

SSIM. Dynamically locates and allocates the ESs. For GPM, SSIM creates a view of the pool of available ESs as a single server, respectively – the network of heterogeneous workstations – as a single metacomputer. Accepts the packets of offspring from GPM in the queue of individuals, ready to be scheduled (QIRSc) and assign each of them to specific ES in accordance with the scheduling policy. Accepts the results of evaluations and forwards them to GPM.

ES. Accepts the packets of offspring from SSIM in the queue of individuals ready to be evaluated (QIRE) and assigns the individuals to separate thread (evaluation thread - ET) for further evaluation. Introducing ET contributes to achieving asynchronous (non-blocking) SSIM – ES interaction by separating the communication from evaluation of the individuals. The queue of results (QR) accumulates the obtained fitness values prior to forwarding them in single batch back to the SSIM.

3.3 Functionality of SSIM

Resource Location and Allocation. The location of the ESs is performed assuming that ESs are installed on each of a priory known workstations in the cluster. The allocation of ES, performed by DCOM is initiated by SSIM during the creating an instance of the remote object. Not only to the issue of locating, creating instance of the object is also closely related to the issues of naming and protecting the communicating entities of the proposed distributed architecture. The benefits of adopting DCOM as a communication paradigm are that the latter, as a true system model uses globally unique identifiers to identify object classes and supported interfaces; encapsulates the life cycle of DCOM-server objects (ESs) via reference counting, and provides a means for secure access to objects and the data they encapsulate.

Scheduling and Load Balancing. The adopted scheduling policy is based on the combination of two strategies as follows. At the stage of evaluating of first batches of current generation SSIM applies static, round robin scheduling (RRS). At this stage SSIM feeds each ESs with 2 batches, providing ES-side overlapping of the evaluation with the communication. Then, in order to deal with eventually different, and dynamically changeable performance characteristics of ESs we propose a Synchronous Incremental Scheduling (SIS) – a dynamic scheduling policy, which allows for minimizing the computational and communicational cost of scheduling yet providing good quality of load balancing. Synchronously with the event of receiving the batch of solutions from ES, the scheduling subsystem of SSIM extracts the first batch from QIRSc and submits it to ES – the source of recently received result. However, even in homogeneous cluster, implementing adopted scheduling policy might not yield to perfect load balancing in case that the total amount of batches cannot be evenly distributed among the active ESs. Implementing batching would only deteriorate the load

balancing due to the increased granularity of the scheduled tasks. In order to estimate the optimal batch size we consider the upper bounds of the efficiency E of load balancing:

$$E = \frac{T_i}{T_a} = \frac{C \quad b/S}{C \quad ceil(b/S)} = \frac{b/S}{ceil(b/S)} \tag{3}$$

where T_i the GP-runtime for evolving single population for "ideal" load balancing when all the individuals are evenly distributed among ESs, T_a is the actual runtime for the best possible load balancing in homogeneous environment implementing batching, C is the computational cost of single batch, b is the amount of scheduled batches, S is the number of active ESs, and `ceil()` is a function that returns the smallest integer greater than or equal a specified value. The amount of batches b is:

$$b = P \quad R_{CO} / s \tag{4}$$

where P is the population size, R_{CO} is the crossover ratio, and s is the batch size. Applying (3) and (4) we obtain the values of E for the following values of the parameters: P=1000 individuals, and R_{CO}=0.8, S=var, and s=var. The values of E are shown in Fig.7. As figure illustrates, increased batch sizes yield lower values of efficiency. To maintain the bound of efficiency of load balancing equal to 1 we propose to dynamically adjust the population size in that the amount of batches, expressed in (4) could be evenly distributed among ESs. The correctness of this approach is based on the empirical observation that the computational effort of GP does not significantly change with small variations in population size. In our approach we consider the variation of population size of 6%, achieved for s=16 individuals as an acceptable in that it does not affect the computational effort of GP. Proposed method for dynamic adjustment of population size is implemented straightforwardly: SSIM includes the amount of currently active ESs in the batch of results, forwarded to GPM. The latter adjusts the population size in accordance with the following rule:

$$P^* = round \quad \frac{P \quad R_{CO}}{s \quad S} \quad \frac{s \quad S}{R_{CO}} \tag{5}$$

where s=16 individuals, R_{CO} and P are the end-user defined crossover ratio and population size respectively.

4. Performance Evaluation

The performance evaluation results are obtained from the developed prototype of MIGP running on local cluster of 10 W2K workstations. The workstations are 450MHz/64MB Pentium II-based Hitachi Flora. They are attached to 100 Base TX Ethernet through Intel 82558-based Ethernet adapter. Workstations are connected in network through Hitachi Summit-48 switching hub. Experimentally obtained speedup characteristics of MIGP are shown in Table 2. The granularity of parallelism derived from these data is about 100. The upper bound of the scalability (assuming that there are no network collisions, and no serialization of the ES-SSIM communications),

calculated as a ratio of the throughput of scheduling subsystem of SSIM to the performance of ES is more than 60, which indicates that one should expect close to linear speedup characteristics of MIGP deployed on the considered cluster of 10 workstations. The experimentally obtained speedup characteristics of MIGP, which are consistent with our anticipation, are shown in Fig.8. The speedup is derived as a ratio of runtime of evolving one generation in serial GP to the runtime of MIGP in configuration where each workstation (including workstation that hosts GPM and SSIM) hosts a single ES.

Table 2. Performance characteristics of MIGP

Characteristic	Value
Batch size, *individuals*	16
Computational cost of evolving one generation of individuals (for $P^*=1000$), *batches*	50
Runtime of evolving one generation in serial implementation of GP, *s*	14
Performance of serial implementation of GP, *batches/s*	3.6
Runtime of evolving one generation on MIGP configured with one remote ES, *s*	15
GPM: throughput of crossover operation, *batches/s*	2800
GPM: Runtime of submitting all the batches to SSIM, *s*	0.14
GPM-SSIM throughput, *batches/s*	370
Throughput of scheduling subsystem of SSIM (SSIM-ES throughput), *batches/s*	260
Performance of ES, *batches/s*	3.8

5. Conclusion

We presented a DCOM-based SSIM for MIGP, aimed to significantly improve the computational performance of GP exploiting the inherent parallelism of GP among the evaluations of individuals. MIGP runs on cost-effective clusters of commodity, non-dedicated, heterogeneous workstations. Developed SSIM represents these workstations as unified virtual resource and addresses the issues of locating and allocating the physical resources, communicating between the distributed entities, scheduling and load balancing. Adopting DCOM as a communicating paradigm offers the benefits of software platform- and network protocol neutrality of proposed implementation and generic support to the issues of naming, locating and allocating of the entities of MIGP. Batching reduces the specific software overhead of DCOM and increases the throughput of SSIM. Developed SIS features reduced runtime overhead and good quality of load balancing. Mechanism of dynamic adjustment of the size of GP-population is proposed as a solution to the problem of load unbalance when scheduled tasks cannot be evenly distributed among the available workstations. Presented results of experimentally obtained speedup characteristics show close to linear speedup of developed MIGP for solving TSIP on cluster of 10 W2K workstations.

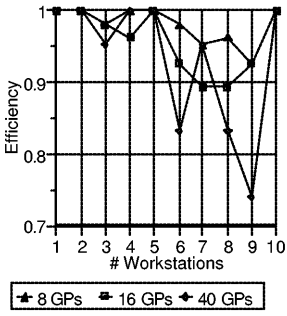

Fig 7. Estimated upper bound of the efficiency of load balancing for different system configurations. Obtained from (4)

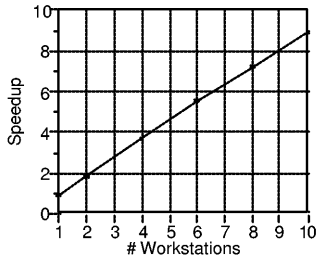

Fig. 8. Experimentally obtained speedup characteristics of MIGP for solving TSIP

References

1. Koza, J.R., Bennett III, F.H., Andre, D., Keane, M.A.: Genetic Programming III: Darwinian Invention and Problem Solving. Morgan Kaufmann Publishers, San Francisco, CA (1999).
2. Buyya, R.: High Performance Cluster Computing, Vol.1, Prentice Hall PTR, Upper Saddle River, New Jersey (1999)
3. Dracopoulos, D.C., Kent, S.: Speeding up genetic programming: A parallel BSP implementation. In: Koza, J.R., Goldberg, D., Fogel, D.B., Riolo, R.L. (eds): Genetic Programming 1996: Proceedings of the First Annual Conference, MIT Press, Stanford University, CA, 28-31 July (1996) 421
4. Oussaidene, M., Chopard, M., Pictet, O., Tomassini, M.: Parallel Genetic Programming and its Application to Trading Model Induction, Parallel Computing 23 (1997) 1183-1198.
5. Tanev, I.T., Uozumi, T., Ono, K.: DCOM-based Parallel Distributed Implementation of Genetic Programming, Parallel and Distributed Computing Practices Journal, Special Issue on Distributed Object Oriented Systems, 1, Vol.3 (2000)
6. Globus – http://www.globus.org/
7. Globe – http://www.cs.vu.nl/~steen/globe/
8. Legion – http://legion.virginia.edu/
9. Condor – http://www.cs.wisc.edu/condor/
10. Ljung L.: System Identification, Theory for the User. Prentice Hall, Englewood Cliffs, New Jersey (1987)
11. Akhmetov, D.F., Dote, Y.: Fuzzy System Identification With General Parameter Radial Basis Function Neural Network, In: Farinwata, S., Filev, D., Langari, R. (Eds.): Analytical Issues in Fuzzy Control: Synthesis and Analysis, John Wiley & Sons, Ltd, United Kingdom (2000) 73-92
12. Microsoft Corporation: COM Specification (1995), http://www.microsoft.com/com/ resources/specs.asp
13. Scholtyssik, K.: NT-MPICH – Project Description (2000) http://www.lfbs.rwth-aachen.de/~karsten/projects/nt-mpich/index.html