MAPPING PARALLEL PROGRAMS INTO HIERARCHICAL DISTRIBUTED COMPUTER SYSTEMS

Victor G. Khoroshevsky and Mikhail G. Kurnosov
Computer Systems Laboratory, A.V. Rzhanov Institute of Semiconductor Physics, 13 Lavrentyev Ave, Novosibirsk, Russia

Keywords: Parallel programs mapping, Task allocation, Task assignment, MPI, Graph partitioning, Distributed computer systems, Multicore computer clusters, Parallel computer systems.

Abstract: In most cases modern distributed computer systems (computer clusters and MPP systems) have hierarchical organization and non-uniform communication channels between elementary machines (computer nodes, processors or processor cores). Execution time of parallel programs significantly depends on how they map to computer system (on what elementary machines parallel processes are assigned and what channels for inter-process communications are used). The general problem of mapping a parallel program into a distributed computer system is a well known NP-hard problem and several heuristics have been proposed to approximate its optimal solution. In this paper an algorithm for mapping parallel programs into hierarchical distributed computer systems based on task graph partitioning is proposed. The software tool for mapping MPI applications into multicore computer clusters is considered. The quality of this algorithm with the NAS Parallel Benchmarks is evaluated.

1 INTRODUCTION

A message passing model became widespread for development of parallel programs for distributed computer systems (CS; for example, MPI and PVM). In this model a parallel program can be presented by a task graph that defines a pattern of communications between parallel processes.

Execution time of parallel programs significantly depends on how they map to CS, on what elementary machines (EM) parallel processes are assigned and what channels for inter-process communications are used.

An objective of optimal mapping of parallel program into distributed CS is to minimize communications costs and load disbalance of EMs.

For distributed CSs with static network structures (hypercube, 3*D*-torus or mesh) and SMP-clusters efficient algorithms for mapping parallel programs are developed (Ahmad, 1997), (Bokhari, 1981), (Lee, 1989), (Yau, 1993), (Yu, 2006), (Chen et al. 2006).

Modern distributed CSs are multiarchitectural (Khoroshevsky, 2005, 2008). Depending on level of consideration of their functional structures, they can look both as MISD, and as SIMD, and as MIMD systems. For such systems hierarchical organization and non-uniform communication channels between EMs are characteristic.

A typical number of levels in modern distributed CSs is vary from 2 up 4 (for example, shared memory of processor cores, shared memory of processors, computer nodes interconnect, links between second stage switches in fat tree topology etc.).

In most popular MPI libraries (MPICH2 and OpenMPI) realized the round robin and the linear algorithms of mapping parallel programs into CSs. This algorithms do not take into account hierarchical organization of modern distributed CSs. The round robin algorithm allocate parallel processes between N EMs in the follow order: first process allocated on first EM, second process on second EM, …, process N on EM N, process $N + 1$ on process 1 and etc. The linear algorithm allocates M processes between first M EMs.

In this paper we consider the problem of optimal mapping parallel programs into hierarchical distributed CS (particularly multicore computer clusters).

A heuristic algorithm of mapping parallel programs is proposed. A software tool for optimization of mapping MPI programs to multicore

Khoroshevsky V. and Kurnosov M. (2009).
MAPPING PARALLEL PROGRAMS INTO HIERARCHICAL DISTRIBUTED COMPUTER SYSTEMS.
In *Proceedings of the 4th International Conference on Software and Data Technologies*, pages 123-128
DOI: 10.5220/0002240601230128

computer clusters is developed. A results of natural experiments on mapping parallel MPI-programs from High-Performance LINPACK (HPL) and NAS Parallel Benchmarks into multicore computer cluster are presented.

The rest of the paper is organized as follows: Section 2 describes some related works. Section 3 gives a description of the problem, and Section 4 describes the algorithm. Section 5 outlines and evaluates the experimental tests. Finally, we summarize our work in Section 6.

2 RELATED WORK

The problem of mapping parallel programs to CS has been well described. There have been many distinct categories of research, each having a different focus. A large part of the work (Kielmann et al. 1999), (Almasi et al. 2005), (Faraj et al. 2005) has concentrated on working with communication network topology graph only while still ignoring task graph structure. In the next category, researchers have worked on communication-sensitive clustering while still ignoring any network topology considerations. The main objective here is the partitioning of task graph into balanced groups while reducing inter-partition communication (Lee, Kim & Park, 1990), (Lopez-Benitez, Djomehri & Biswas, 2001). The graph partitioning algorithm (Karypis, Kumar, 1999), (Hendrickson, Leland, 1995) is widely used in the MPI performance optimization (Träff, 2002). It requires the task graph to describe the communication behavior of the program, which could be derived from trace or user input (Chen et al. 2006).

In this paper we developed a heuristic algorithm for mapping parallel programs into hierarchical distributed CSs. The algorithm for working at both a task graph and the information about communication network hierarchy.

3 THE MAPPING PROBLEM

Let's hierarchical distributed CS has N homogeneous elementary machines and a communication network with hierarchical organization. Such a communication network can be described by a tree with L levels. Each level of system is formed by own type of functional modules (for example, telecommunication racks, computer nodes, processors etc.) which interconnected via communication channels of current level. In Figure 1 an example of hierarchical CS, three nodes computer cluster, is shown.

For CS description following denotations are accepted: n_l – is a number of elements placed at level $l \in \{1, 2, \ldots, L\}$; n_{lk} – is a number of children of element $k \in \{1, 2, \ldots, n_l\}$ at level l; $g(l, k_1, k_2)$ – is a number of the element level, which is the lowest common ancestor for elements $k_1, k_2 \in \{1, 2, \ldots, n_l\}$; b_l – is a bandwidth of communication channels at level l ($[b_l]$ = bit/sec.); C_{lk} – is a set of elementary machines belonging to the descendants of element k at level l; $c_{lk} = |C_{lk}|$; $C_{11} = C$; $C = \{1, 2, \ldots, N\}$.

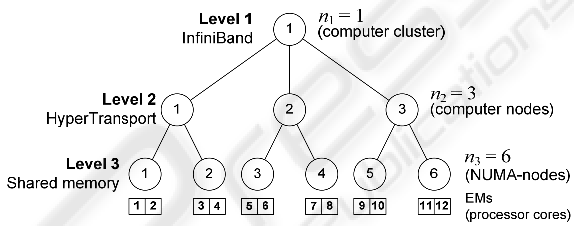

Figure 1: Cluster: 3 nodes 2 x AMD Opteron 275 ($N = 12$; $L = 3$; $n_{23} = 2$; $C_{23} = \{9, 10, 11, 12\}$; $g(3, 3, 4) = 2$; $z(1, 7) = 1$).

A message-passing parallel program is represented by a weight undirected task graph $G = (V, E)$. The vertices $V = \{1, 2, \ldots, M\}$ correspond to parallel processes and the edges $E \subset V \times V$ represent communications between the processes. Weight d_{ij} of edge $(i, j) \in E$ is a number of bytes transmitted between processes i and j during program execution ($[d_{ij}]$ = bytes). We assume that $M \le N$.

3.1 Estimation of Parallel Program Execution Time

Formally, the problem of optimal mapping of parallel program into distributed hierarchical CS is to find injective function $f: V \to C$, which maps parallel processes to EMs. It is required to find x_{ij}:

$$X = \{x_{ij} : I \in V, j \in C\}, \quad x_{ij} = \begin{cases} 1, & \text{if } f(i) = j; \\ 0 & \text{else.} \end{cases}$$

We use the expected execution time t of parallel program as an optimization criterion. The execution time t of parallel program is defined as maximum from its processes execution times.

The execution time t_i of parallel process $I \in V$ includes a time of computations and a time of communications with adjacent processes. We take into account the communication costs only due to the homogeneity of EMs. Then

$$t = \max_{i \in V}\{t_i\} = \max_{i \in V}\left\{\sum_{j=1}^{M}\sum_{p=1}^{N}\sum_{q=1}^{N} x_{ip} \cdot x_{jq} \cdot t(i,j,p,q)\right\},$$

where $t(i, j, p, q) = d_{ij} / b_{z(p, q)}$ is a time of communications between processes $i, j \in V$, which are allocated to EMs p and q, correspondingly ($p, q \in C$). The function $z(p, q)$ sets up a correspondence between machines p and q, and the number of communication network level through which they interact. In Figure 1 function $z(1, 7) = 1$, because the processor cores 1 and 7 belong to different computer nodes which interact via InfiniBand network.

3.2 Optimization Problem

Let's formulate the problem of optimal mapping of parallel program into hierarchical distributed CS with the injectivity of the function f taken into account:

$$T(X) = \max_{i \in V}\left\{\sum_{j=1}^{M}\sum_{p=1}^{N}\sum_{q=1}^{N} x_{ip} \cdot x_{jq} \cdot t(i,j,p,q)\right\} \to \min_{(x_{ij})} \quad (1)$$

subject to:

$$\sum_{j=1}^{N} x_{ij} = 1, \quad i = 1,2,\ldots,M, \quad (2)$$

$$\sum_{i=1}^{M} x_{ij} \le 1, \quad j = 1,2,\ldots,N, \quad (3)$$

$$x_{ij} \in \{0,1\}, \quad i \in V, \ j \in C. \quad (4)$$

The constraints (2), (4) ensure that each process allocated on one EM, constraints (3) guarantee that each EM execute one process.

4 THE MAPPING ALGORITHM

The problem (1) – (4) is a discrete optimization problem. The heuristic algorithm TMMGP (Task Map Multilevel Graph Partitioning) for solving the problem is developed. The algorithm based on multilevel procedure of k-way graph partitioning. Let's formulate the last problem.

4.1 Graph Partitioning Problem

The k-way graph partitioning problem is defined as follows: given a graph $G' = (V', E')$ with $V' = \{1, 2, \ldots, n\}$, partition V' into k disjoint subsets $V'_1, V'_2, \ldots, V'_k$ such that $V'_1 \cap V'_2 \cap \ldots \cap V'_k = \varnothing$, $V'_1 \cup V'_2 \cup \ldots \cup V'_k = V'$ and the maximal sum of the edge-weights incident to any subset is minimized. Let

$$E'(i,j) = \{(u,v) \in E' : u \in V'_i, v \in V'_j, i \ne j\}$$

denote a set of edges whose incident vertices belong to subsets V'_i and V'_j. The function

$$c(u, v, i, j) = w(u, v)W(i, j)$$

is an edge-weight which incident to different subsets; $w(u,v)$ is a weight of the edge $(u, v) \in E'$, $W(i, j)$ is an additional weight for edges incident to subsets i and j.

The formal problem statement of optimal k-way graph partitioning with constraints for V'_i taken into account is presented below.

$$F(V'_1, V'_2, \ldots, V'_k) = \max_{i=1,k}\left\{\sum_{j=1}^{k} \sum_{(u,v) \in E'(i,j)} c(u,v,i,j)\right\} \to \min_{(V'_1, V'_2, \ldots, V'_k)} \quad (5)$$

subject to:

$$V'_1 \cap V'_2 \cap \ldots \cap V'_k = \varnothing, \quad (6)$$

$$V'_1 \cup V'_2 \cup \ldots \cup V'_k = V', \quad (7)$$

$$|V'_i| > 0, \quad i = 1,2,\ldots,k, \quad (8)$$

$$|V'_i| \le s, \quad i = 1,2,\ldots,k. \quad (9)$$

4.2 The Mapping Algorithm

The mapping algorithm TMMGP consists of the following steps.

1) For the task graph $G = (V, E)$ is solved the problem (5) – (9) – the graph is partitioned into $k = [(M - 1) / c_{L1}] + 1$ disjoint subsets $V'_1, V'_2, \ldots, V'_k$ with values: $s = c_{L1}$, $c(u, v, i, j) = d_{uv} / b_{g(L, i, j)}$ (see Section 3).

2) The mapping $f : V \to C$ is built as follows. The processes from subset V'_i is allocated to EMs from set C_{Li}.

The multilevel heuristic algorithms for graph partitioning (Hendrickson, Leland, 1995), (Karypis, Kumar, 1999) became widely spread in practice. Such algorithms allow us to find approximate solutions for problem (5) – (9) in an acceptable time.

In this paper at step 1 we used the multilevel graph partitioning algorithm introduced in (Karypis, Kumar, 1999). The complexity of the algorithm is $O(|E|\log_2 k)$.

In Figure 2 a pseudocode of the TMMGP algorithm is shown.

<u>Input</u>: a task graph $G = (V, E)$; a CS description: $N, L, n_l, n_{lk}, C, C_{lk}, c_{lk}; M'$.

<u>Output</u>: mapping $x[j, c]$, $x[j, c] = 1$ if process j allocated to EM c, and $x[j, c] = 0$ otherwise.

1 $k \leftarrow [(M - 1) / c_{L1}] + 1$
2 $s \leftarrow c_{L1}$
3 $M \leftarrow \log_2(M / M') + 1$
4 **for** $i \leftarrow 1$ **to** m **do**
5 $\quad G_i \leftarrow CoarseGraph(G_{i-1})$
6 **end for**
7 $P_m \leftarrow RecursiveBisection(G_m, k, s)$
8 **for** $i \leftarrow m$ **to** 1 **do**
9 $\quad P_{i-1} \leftarrow ProjectPartition(G_{i-1}, P_i)$
10 $\quad P_{i-1} \leftarrow RefinePartition(G_{i-1}, P_{i-1})$
11 **end for**
12 **for** $j \leftarrow 1$ **to** M **do**
13 $\quad c \leftarrow Dequeue(C[P_{0,j}])$
14 $\quad x[j, c] \leftarrow 1$
15 **end for**

Figure 2: A pseudocode of the TMMGP algorithm.

In the lines 4 – 6 a sequence of smaller graphs $G_1, G_2, \ldots, G_m$ is built. The function *CoarseGraph* coarse the graph G_i to smaller graph G_{i+1} by stochastic algorithm Heavy Edge Matching (Karypis, Kumar, 1998), $|V_{i+1}| \approx |V_i| / 2$, $G_0 = G$. The function *RecursiveBisection* realize the recursive bisection algorithm LND (Schloegel et. al., 2003) of small graph G_m into k subsets with constrain $|V_i'| \le s$. The function *ProjectPartition* map the partition P_i of the graph G_i to the graph G_{i-1}. The function *RefinePartition* implements the FM heuristic algorithm of partition refinement (Fiduccia, Mattheyses, 1982). In the line 13 $C[k]$ is a queue of EMs from C_{Lk} set and $P_{0,j}$ is a number of subset of partition P_0 which process j belong to.

The parameter M' is a number of vertices in graph G_m. The value of M' chooses such that recursive bisection implements fast (usually on practice $M' \le 10 \cdot k$).

A computational complexity of the TMMGP algorithm is $O(|E|\log_2 k + M)$.

The example of task graph mapping into computer cluster by the TMMGP algorithm is shown in Figure 3.

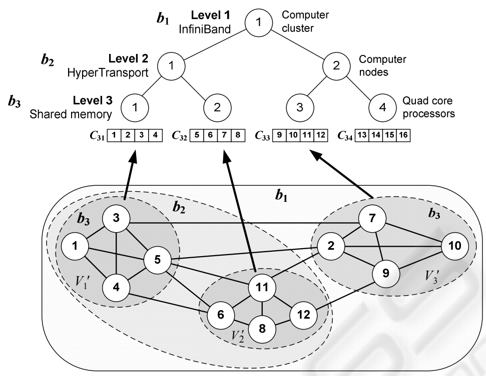

Figure 3: Example of mapping task graph by TMMGP algorithm ($N = 16$; $L = 3$; $M = 12$; $b_1 = 2$ GBps; $b_2 = 6$ GBps; $b_3 = 8$ GBps).

5 EXPERIMENTS

The software tool MPITaskMap for optimization of mapping MPI programs into multicore computer clusters is created.

5.1 Mapping Tools

The MPITaskMap tool consists of the following three components (see Figure 4):

1) OTFStat is the tool for analyzing traces of MPI programs in Open Trace Format (OTF) (Knüpfer et al. 2006) and building task graphs.

2) CommPerf is the tool for benchmarking performance of communication channels between processor cores of computer cluster.

3) MPITaskMap is the tool for mapping MPI programs into processor cores. This component gets MPI program's task graph and system description as an input and builds a script for launching the program with optimized mapping.

All components are implemented in ANSI C for GNU/Linux operation system.

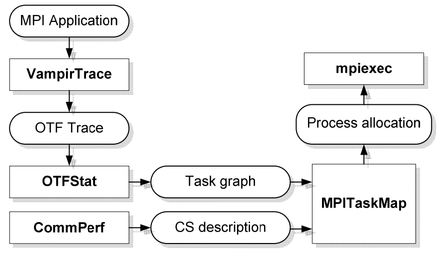

Figure 4: Process of the mapping MPI applications.

5.2 Experiment Organization

We used MPI programs from packages NAS Parallel Benchmarks (NPB) and HPL in our experiments. The structures of task graphs of HPL, Conjugate Gradient (CG) and NPB Multigrid (MG) benchmarks are shown in Figure 5, 6 and 7, correspondingly. All graphs are built by OTFStat tool.

Computers clusters of the following configurations are used in experiments:

– cluster Xeon16: 4 nodes (2 x Intel Xeon 5150) interconnected via Gigabit/Fast Ethernet networks;

– cluster Opteron10: 5 nodes (2 x AMD Opteron 248) interconnected via Gigabit/Fast Ethernet networks.

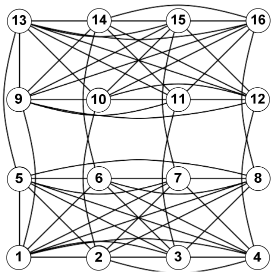

Figure 5: HPL benchmark task graph (16 processes, PMAP=0, BCAST=5).

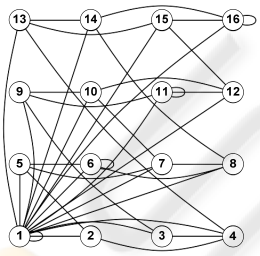

Figure 6: NPB Conjugate Gradient task graph (16 processes, CLASS B).

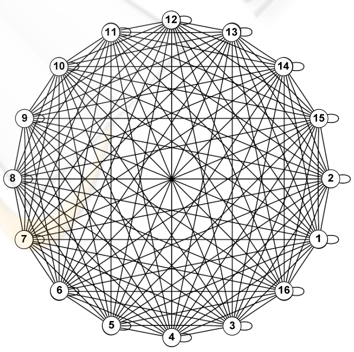

Figure 7: NPB Multigrid task graph (16 processes, CLASS B).

5.3 Results

The execution times of MPI benchmarks with mapping into cluster Xeon16 by round robin algorithm ($T(X_{RR})$, default algorithm of mpiexec tool) and by TMMGP algorithm ($T(X_{TMMGP})$) are presented in Table 1.

The communication network of Xeon16 consists of two levels. The first level is a Gigabit / Fast Ethernet network between nodes, the second level is a shared memory of processors inside nodes.

The working time of algorithm TMMGP on Intel Core 2 Duo 2.13GHz processor did not exceed $5\cdot10^{3}$ sec. for all benchmarks.

As we can see from results, the quality of mapping significantly depends on a task graph structure. The optimization of mapping of parallel programs with non-uniform task graphs (for example, NPB CG or HPL) can considerably reduce its execution time.

Table 1: Experimental results.

Cluster interconnect	$T(X_{RR})$, sec.	$T(X_{TMMGP})$, sec.	$T(X_{RR})$ / $T(X_{TMMGP})$
High Performance Linpack			
Fast Ethernet	1108,69	911,81	1,22
Gigabit Ethernet	263,15	231,72	1,14
NPB Conjugate Gradient			
Fast Ethernet	726,02	400,36	1,81
Gigabit Ethernet	97,56	42,05	2,32
NPB Multigrid			
Fast Ethernet	23,94	23,90	1,00
Gigabit Ethernet	4,06	4,03	1,00

It is necessary to notice, what a facilities on optimization of mapping parallel programs with full task graphs (for example, NPB Multigrid) sufficiently limited. Also, the quality of mapping depends on difference in performance of communication channels on several levels.

6 CONCLUSIONS

In this paper the problem of optimal mapping parallel MPI programs into multicore computer clusters is considered. The heuristic algorithm TMMGP of mapping parallel programs is proposed.

The algorithm working with both the task graph and the information about communication network hierarchy. Software tool for optimization of mapping MPI programs is developed. Examples of mapping parallel programs from NAS Parallel Benchmarks are presented.

The feature work is to integrate the mapping algorithm into resource management systems.

ACKNOWLEDGEMENTS

This work was supported by Russian foundation for basic research under grants 08-07-00018, 07-07-00142, 08-07-00022, 08-08-00300, 09-07-90403 and grant Support of Russian Leading Scientific School no. 2121.2008.9.

REFERENCES

Ahmad, I., 1997. A Parallel Algorithm for Optimal Task Assignment in Distributed Systems. In *Proceedings of the Advances in Parallel and Distributed Computing Conference*, pp. 284.

Bokhari, S. H., 1981. On the mapping problem. *IEEE Transactions on Computers,* Vol. 30, №3, pp. 207-214.

Lee, C., 1989. On the mapping problem using simulated annealing. In *Proceedings of Computers and Communications*, pp. 40-44.

Yau, S., 1993. A task allocation algorithm for distributed computing systems. In *Proceedings of Computer Software and Applications Conference*, pp. 336-342.

Yu, H., 2006. Topology Mapping for Blue Gene/L Supercomputer. In *Proceedings of ACM/IEEE Conference Supercomputing*, pp. 52.

Kielmann, T., Hofman, R.F.H., Bal, H.E., Plaat, A. and Bhoedjang, R., 1999. MagPIe: MPI's collective communication operations for clustered wide area systems. In *ACM SIG-PLAN Notices* 34, pp. 131-140.

Almási, G., Heidelberger, P., Archer, C. J., Martorell, X., Erway, C. C., Moreira, J. E., Steinmacher-Burow, B., and Zheng, Y., 2005. Optimization of MPI collective communication on BlueGene/L systems. In *Proceedings of the 19th Annual international Conference on Supercomputing*, pp. 253-262.

Faraj, A. and Yuan, X., 2005. Automatic generation and tuning of MPI collective communication routines. In *Proceedings of the 19th Annual international Conference on Supercomputing*, Cambridge, Massachusetts, pp. 393 – 402.

Lee, C. H., Kim, M., and Park, C. I., 1990. An efficient K-way graph partitioning algorithm for task allocation in parallel computing systems. In *Proceedings of the First international Conference on Systems integration on Systems integration,* IEEE Press, pp. 748-751.

Lopez-Benitez, N., Djomehri, M. J., and Biswas, R., 2001. Task Assignment Heuristics for Distributed CFD Applications. In *Proceedings of the 2001 international Conference on Parallel Processing Workshop*. IEEE Press., p. 128.

Karypis, G., Kumar, V., 1999. A fast and high quality multilevel scheme for partitioning irregular graphs. In *SIAM Journal on Scientific Computing*, Vol. 20, No. 1, pp. 359-392.

Hendrickson, B., Leland, R., 1995. A multilevel algorithm for partitioning graphs. In *Proceedings of the ACM/IEEE conference on Supercomputing*.

Träff, J. L., 2002. Implementing the MPI process topology mechanism. In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*. IEEE Press., pp. 1 - 14.

Chen, H., Chen, W., Huang, J., Robert, B., and Kuhn, H., 2006. MPIPP: an automatic profile-guided parallel process placement toolset for SMP clusters and multiclusters. In *Proceedings of the 20th Annual international Conference on Supercomputing,* pp. 353-360.

Khoroshevsky, V.G., 2008. *Computer Systems Architecture*, MGTU, Moscow [in Russian].

Karypis, G., 1998. Multilevel k-way partitioning scheme for irregular graphs. In *Journal of Parallel and Distributed computing*, Vol. 48, pp. 96-129.

Schloegel, K., G. Karypis, V. Kumar, 2003. Graph partitioning for high-performance scientific simulations. In *Sourcebook of parallel computing*, pp. 491-541.

Fiduccia, C. M., R. M. Mattheyses, 1982. A linear-time heuristic for improving network partitions. In *Proc. of conference "Design Automation"*, pp. 175-181.

Khoroshevsky, V.G., Mamoilenko, S.N. Maidanov, Y.S., Sedelnikov, M.S., 2005. Space-distributed multicluster computer system with multiprogramme regimes supporting. In *Proceedings of the Second IASTED International Multi-Conference on Automation, Control and Information Technology*. Software Engineering, ASTA Press.

Khoroshevsky, V.G., Mamoilenko, S.N., Kurnosov, M.G., Medvedeva, N.A., 2006. Space-Distributed Multi-Cluster Computer System for Training in Parallel Computational Technologies. In *Proc. of 7th International Workshop and Tutorials on Electron Devices and Materials,* pp. 218-219.

Knüpfer, A., Brendel, R., Brunst, H., Mix, H., Nagel, W.E., 2006. Introducing the Open Trace Format (OTF). In *Vassil N. Alexandrov, Geert Dick van Albada, Peter M. A. Sloot, Jack Dongarra (Eds): Computational Science - ICCS 2006: 6th International Conference,* pp. 526-533.