UDC 004.94:519.6

D. K. ANDREICHENKO, D. V. MELNICHUK, A. A. EROFTIEV

Saratov State University, Russia

DYNAMIC BALANCING OF COMPUTING LOAD IN HIGHLY PARALLEL PROBLEM SOLVING

Optimizing the algorithms for parallel computing systems requires effective dynamic load balancing. OpenMP and MPI technologies allow for software development for the most of modern parallel computing architectures but MPI does not provide standard facilities for dynamic load balancing. In this paper, we propose MPI-MAP parallel programming pattern, which solves that problem by implementing MAP stage of well-known MAP-REDUCE pattern with standard facilities of MPI. The efficiency of dynamic load balancing with OpenMP and MPI-MAP is shown on symmetric multiprocessing systems with shared memory, massive parallel processing clusters with distributed memory and computing systems with Intel Xeon Phi co-processors. As an example, the problem of numerical modeling the influence of typical nonlinearities on output vector functions of nonlinear stabilization system of mobile control object is solved.

Key words: *MPI, OpenMP, parallel programming, hybrid dynamical system, mathematical modeling, HPC (high performance computing).*

Introduction

The number of methods [1] and corresponding metrics for estimating the energy efficiency of modern parallel computing system's hardware is known but these are not versatile. Finding metrics for estimating the energy efficiency of software is even more complicated [2]. Despite this, the acceleration factor of parallel program [3] can be used as an analog of power metric when developing green software for modern parallel computer architectures, which is classically provided that the asymptotic complexity of initial algorithms is minimal. Minimizing the asymptotic complexity of algorithms may involve using adaptive techniques so that dynamic load balancing based on "manager-workers" strategy [3] is required. Parallel programming technologies such as OpenMP and MPI can be used for developing the software for most modern parallel computer architectures. "Manager-workers" strategy is part of OpenMP standard in versions 3.x and 4.x [4] but it is limited to multiple threads within one process in an operating system (OS) running on a single node of massive parallel processing (MPP) cluster. That strategy is not implemented within modern MPI standard [5], which is traditionally used on MPP clusters, but it can be substituted with the MPI-MAP pattern [6] proposed by the authors of this paper. MPI-MAP implements the MAP stage of the MAP-REDUCE pattern [7] with standard MPI process interchange operations to balance the computational load dynamically according to "manager-workers" strategy. MPI-MAP is very efficient when time that program runs in single worker is much more than time needed for data interchange within MPI processes. For example, MPI-MAP can be used in parallel algorithms of mathematical modeling of controlled dynamic systems containing interacting objects with parameters concentrated in space as well as objects with parameters distributed in space, based on a body of hybrid dynamical systems (HDS) [8]. HDS is a mathematical model in form of system of ordinary differential equations (ODE) and partial differential equations (PDE) combined with boundary and relation conditions in presence of given initial conditions. The primary theorems on the stability of linear and linearizeable HDS are formulated and proven in [8]. Various modifications of parametric synthesis (the algorithm of picking feedback parameters to sustain required quality of transient processes) are shown in [9-11]. Nevertheless, the question of comparing the efficiency of dynamic load balancing based on OpenMP and MPI-MAP stays open, in case of running those mathematical models on SMP systems as well as MPP clusters and energy efficient co-processors such as Intel Xeon Phi.

1. Formulation of the study

The numeric modeling of output vector functions of initial nonlinear HDS should be held after parametric synthesis on the linearized model. This numeric modeling includes non-dimensional parameters which characterize the impact of typical nonlinearities and change discretely within specified ranges. That problem is highly parallel but it requires dynamic balancing of computing load. We here formulate the problem of

comparing the efficiency of dynamic load balancing based on OpenMP and MPI-MAP when modeling the impact of typical nonlinearities on output vector functions of the HDS on SMP systems, MPP clusters and Intel Xeon Phi co-processors. The test model is the mathematical model of the nonlinear stabilization system of steerable control object [6] (e.g. the missile granting the deformation of its structure). The parametric synthesis had been held earlier in [11].

2. MPI-MAP pattern

The well-known MAP-REDUCE pattern of parallel programming allows for parallelization of various data-processing algorithms. It consists of 2 stages. On a first stage called MAP some sufficiently long finite sequence is being transformed into another finite sequence element-wise:

$$\{x_n\}_{n=1}^{N} \to \{y_n\}_{n=1}^{N},\ y_n = f(x_n);\quad x_n \in X,\ y_n \in Y,\ f: X \to Y. \tag{1}$$

On a second stage called REDUCE the result sequence is being reduced to one value:

$$y = y_1\,\mathrm{Op}\,y_2\,\mathrm{Op}\ldots\mathrm{Op}\,y_N,\ y \in Y, \tag{2}$$

so that Op operation is associative. The MAP stage is effectively parallel since the sequence is processed element-wise. The REDUCE stage is also effectively parallel because Op is associative. The dynamic parallel load balancing is required if time needed for operations f and Op can change heavily depending on input.

In the MPI standard, the REDUCE stage is implemented with MPI_Reduce operation. The MAP stage can be implemented with another standard MPI functions. However, MPI doesn't offer the dynamic load balancing by itself. The proposed MPI-MAP pattern implementation [6] with dynamic load balancing implies that some MPI process (e.g. process with rank 0) acts as a manager and all other processes $j = \overline{1, N_w}$, $N_w \ll N$ are workers. MPI-MAP pattern pseudocode goes below.

I. For $j = \overline{1, N_w}$ **:**

1) Process j: start blocking receive (MPI_Recv) x_j
2) Process 0: *a)* start buffered transfer (MPI_Bsend) x_j to process j; *b)* start non-blocking receive (MPI_Irecv) y_j from process j;
3) Process j: *a)* finish blocking receive (MPI_Recv) x_j; *b)* start calculation of $y_j = f(x_j)$.

II. For $k = \overline{N_w + 1, N}$ **:**

1) Process 0: start blocking wait for receive (MPI_Waitany) y_ν, $1 \le \nu \le N$ from any process j, $1 \le j \le N_w$
2) Process j, $1 \le j \le N_w$: *a)* finish calculation of $y_\nu = f(x_\nu)$, $1 \le \nu \le N$; *b)* start buffered send (MPI_Bsend) y_ν, $1 \le \nu \le N$, to process 0; *c)* start blocking receive (MPI_Recv) x_k from process 0;
3) Process 0: *a)* finish blocking wait for receive (MPI_Waitany) y_ν, $1 \le \nu \le N$ from some process j, $1 \le j \le N_w$; *b)* start buffered send (MPI_Bsend) x_k to process j, $1 \le j \le N_w$; *c)* start non-blocking receive (MPI_Irecv) y_k from process j, $1 \le j \le N_w$;
4) Process j, $1 \le j \le N_w$: *a)* finish blocking receive (MPI_Recv) x_k; *b)* start calculation of $y_k = f(x_k)$

III. N_w times run:

1) Process 0: start blocking wait for receive (MPI_Waitany) y_ν, $1 \le \nu \le N$ from any process j, $1 \le j \le N_w$
2) Process j, $1 \le j \le N_w$: *a)* finish calculation of $y_\nu = f(x_\nu)$, $1 \le \nu \le N$; *b)* start buffered send (MPI_Bsend) y_ν to process 0; *c)* go to stand-by mode after start of blocking receive of input data (MPI_Recv);
3) Process 0: finish blocking wait for receive (MPI_Waitany) y_ν, $1 \le \nu \le N$ from some process j, $1 \le j \le N_w$.

Intercommunications between tasks run by workers are supposed to be minimal or absent at all. It is also supposed that run time of calculation $y_n = f(x_n)$ is much more than time needed to send data x_n and y_n.

3. Modeling the influence of typical nonlinearities in the HDS

The block diagram of an HDS with piecewise continuous input vector function $\mathbf{x}(t)$ and continuous output vector function $\mathbf{y}(t)$ of time t, $\mathbf{x}: \mathbb{R} \to \mathbb{R}^{N_x}$, $\mathbf{y}: \mathbb{R} \to \mathbb{R}^{N_y}$ (fig. 1) fits model equations similar to [6]:

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{h}, \boldsymbol{\mu});\quad \dot{\mathbf{u}} = \mathbb{F}(\mathbf{u}, \mathbf{x}, \mathbf{y}, \dot{\mathbf{y}}, \boldsymbol{\mu}),\ \mathbf{r} \in \Omega;$$
$$\mathbb{G}(\mathbf{u}, \mathbf{y}, \boldsymbol{\mu})\big|_S = 0;\quad \mathbf{h} = \int_S \mathbb{H}(\mathbf{u}, \boldsymbol{\mu})\,dS; \tag{3}$$
$$\mathbf{y}(0) = \mathbf{y}_0,\ \mathbf{u}(\mathbf{r}, 0) = \mathbf{u}_0(\mathbf{r}).$$

Here $\mathbf{r} \in \mathbb{R}^{N_r}$ is a set of independent space coordinates assigned to some individual point of the object with space-distributed parameters, $\Omega \subset \mathbb{R}^{N_r}$ is an area occupied by objects with space-distributed parameters, $S = \partial\Omega$ is a domain boundary, $\mathbf{u}(\mathbf{r}, t)$,

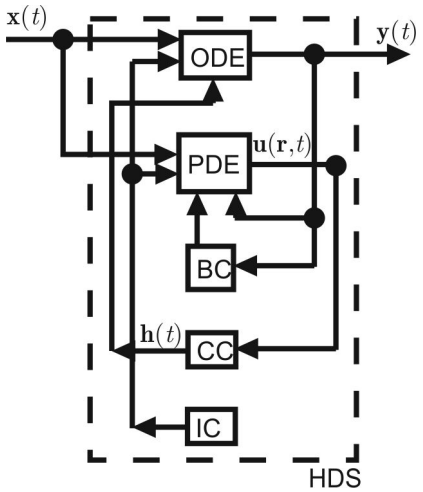

Fig. 1. The block diagram of the HDS

$\mathbf{u}: \mathbb{R}^{N_r} \times \mathbb{R} \to \mathbb{R}^{N_u}$ is a distributed output vector function, $\mathbf{f}: \mathbb{R}^{N_x} \times \mathbb{R}^{N_y} \times \mathbb{R}^{N_h} \times \ \times \mathbb{R}^{N_\mu} \to \mathbb{R}^{N_y}$, differential operators $\mathbb{F}$, $\mathbb{G}$, $\mathbb{H}$ containing partial derivatives $\partial/\partial\mathbf{r}$, $\partial^2/\partial\mathbf{r}^2$,..., fit equations in partial derivatives, boundary and relation conditions; $\boldsymbol{\mu} \in \mathbb{R}^{N_\mu}$ are parameters characterizing the influence of typical nonlinearities; dot on top of the symbol means differentiation by time t. The linearization of model equations (3) corresponds to the situation when $\boldsymbol{\mu} = 0$. Here the parallel algorithm of the parametric synthesis [11] actually uses MPI-MAP pattern to transform the set of frequency range boundaries to integrals over frequency ranges.

After performing the parametric synthesis on linearized model of controlled HDS ($\boldsymbol{\mu} = 0$) the numeric modeling of output vector functions of the initial nonlinear controlled HDS ($\boldsymbol{\mu} \neq 0$) should be held to study the influence of typical nonlinearities. The numerical solution of initial boundary value problem (3) is found with fixed input vector function $\mathbf{x}(t)$ and variable $\boldsymbol{\mu} \in \mathbb{R}^{N_\mu}$. At the same time, the components of the vector $\boldsymbol{\mu}$ vary with fixed pitch within some parallelepiped, i.e.

$$\mu_j = k\Delta\mu_j,\ k = \overline{0, N_j},\ j = \overline{1, N_\mu}. \qquad (4)$$

For any fixed value of $\boldsymbol{\mu}_j, j = 1,2,...$ the numerical solution of the initial boundary value problem (3) can be found independently, i.e. in parallel using adaptive methods of numerical integration. Therefore, the MPI-MAP pattern could be used to hold the transformation

$$\boldsymbol{\mu}_j \to \mathbf{y} = \mathbf{y}(t;\boldsymbol{\mu}_j) \qquad (5)$$

having other structural parameters and input vector function fixed. The REDUCE stage is not of interest here. For small and moderate $|\boldsymbol{\mu}_j|$ the function $\mathbf{y}(t;\boldsymbol{\mu}_j)$ will be just slightly different from the function $\mathbf{y}(t;0)$. Here the MPI-MAP pattern could be used to transform the sequence of $\boldsymbol{\mu}_j$, $j = 1,2,3,...$ to the sequence of values which characterize maximum and standard deviation of function $\mathbf{y}(t;\boldsymbol{\mu}_j)$ from function $\mathbf{y}(t;0)$ when $t \in [0, t_{max}]$, $t_{max} \gg 1$, unlike (5).

$$\boldsymbol{\mu}_j \to (v_1, v_2)^T, v_1 = \max_{0 \le t \le t_{max}} |\mathbf{y}(t;\boldsymbol{\mu}_j) - \mathbf{y}(t;0)|,$$
$$v_2 = \left(t_{max}^{-1} \int_0^{t_{max}} |\mathbf{y}(t;\boldsymbol{\mu}_j) - \mathbf{y}(t;0)|^2 \, dt \right)^{1/2}. \qquad (6)$$

Based on the Galerkin projection method using representation

$$\mathbf{u}(\mathbf{r}, t) \approx \sum_{k=1}^{N_\Omega + N_S} u_k(t) \mathbf{W}_k(\mathbf{r}), \qquad (7)$$

where the system of functions $\mathbf{W}_k(\mathbf{r})$, $\mathbf{W}_k : \Omega \to \mathbb{R}^{N_u}$ $k = 1,2,...$ is complete in Ω similarly to [6], the solution of the initial boundary value problem $\Omega \subset \mathbb{R}^{N_r}$ reduces to the numerical solution of the Cauchy problem for the system of ordinary differential equations

$$\dot{\mathbf{Y}} = \mathbf{F}(t, \mathbf{Y}, \boldsymbol{\mu}),\ \mathbf{Y} = (y_1, ..., y_{N_y}, u_1, ..., u_{N_\Omega})^T \qquad (8)$$

using rigidly stable adaptive implicit BDF method [12] with variable pitch and order. Here (8) is numerically integrated independently, i.e. in parallel for various values of $\boldsymbol{\mu}$. BDF method implementation requires the computation of Jacobian $\partial\mathbf{F}(t, \mathbf{Y}, \boldsymbol{\mu})/\partial\mathbf{Y}$ of right sides of ODE system (8). In paper [6] we propose the fast algorithm for such a computation based on quantization of disturbed movement equations of HDS using a variant of Galerkin projection method analogous to a variant of that method used for the quantization of initial nonlinear mathematical model of the HDS. The infinitesimals can be thrown away here in order to reduce computation complexity greatly.

4. Testing the efficiency of parallel computing on symmetric multiprocessor system

Consider the efficiency of parallel computing in modeling the influence of typical nonlinearities on output vector functions of the nonlinear system intended to stabilize the mobile control object (e.g. the missile subject to deformation of its body, see fig. 2). The input vector function includes the components of outer disturbing force $\mathbf{x}(t) = (F_{ey0}(t), F_{ez0}(t))^T$. Vertical deflection angles of body 1 and missile header (body 2) are components of the output vector function $\mathbf{y}(t) = (\beta_{1,3}(t), \beta_{2,3}(t), \beta_{1,2}(t), \beta_{2,2}(t))^T$. The influence of typical nonlinearities is modeled with set of parameters $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3)^T$. Nonlinear model equations for movement of the mobile control object, particularly the explicit expressions for function $\mathbf{f}$ and operators $\mathbb{F}$, $\mathbb{G}$,

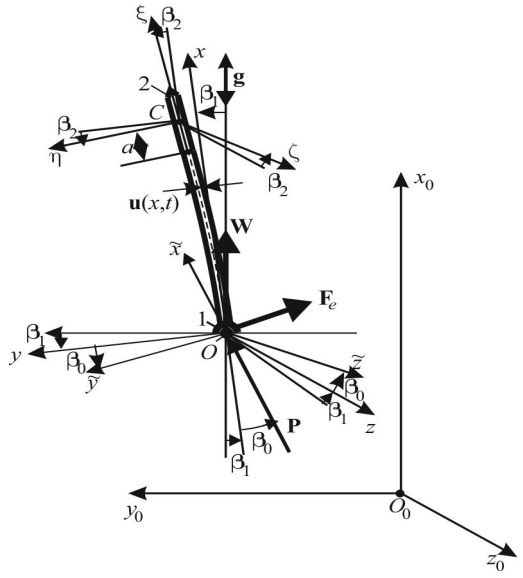

Fig. 2. The design model of mobile control object

$\mathbb{H}$ (3) are represented in [6] and are not included here. As follows from the results represented on fig. 3 for components of output vector functions of nonlinear and linearized models of the stabilization system, the nonlinear controlled HDS is stabilized successfully with prior parametric synthesis [11] held on the linearized model.

The components of the input vector function are set as $F_{e_{y0}}(t) = 1(t)$, $F_{e_{z0}}(t) = 1(t) - 1(t-1)$, where $1(t)$ is a Heaviside step function. Two sets of constructive parameters was used (see [6], p. 110).

In a qualitative sense, the behavior of output vector function of the nonlinear controlled HDS is similar to the behavior of output vector function of the linearized model. At the same time, the analysis of values of maximum and standard deviation (6) shows that the parameter μ_3 [6] characterizing the nonlinearity of saturation type makes the largest impact on output vector functions of the nonlinear stabilization system.

The data represented on fig. 4 characterizes the efficiency of parallel computing using MPI-MAP pattern with 8 worker processes and standard facilities of OpenMP technology in finding the maximum and standard deviations (6) when $\mu_1 \in [0, 0.05]$, $\mu_2 \in [0, 0.1]$, $\mu_3 \in [0, 0.08]$ for the first parameter set and $\mu_1 \in [0, 0.05]$, $\mu_2 \in [0, 0.1]$, $\mu_3 \in [0, 0.065]$ for the second parameter set, respectively. The spatial grids of parameters $\boldsymbol{\mu}$ consisted of 5 nodes for variables μ_1 and μ_2 and 9 nodes for variable μ_3. The calculations were performed on a PC with 4-core Intel Core i7 3610 QM CPU with Hyper-threading enabled.

As seen from the results above, MPI-MAP successfully competes with OpenMP standard library in managing the task sets and dynamic balancing of computing load at small number of workers. The superlinear acceleration of a parallel program may be explained with better utilization of CPU cache.

5. Testing the efficiency of MPI-MAP pattern on SMP cluster

Fig. 5 represents data which characterizes the acceleration of MPI-MAP based computing depending on the number of the workers. The calculation of maximum

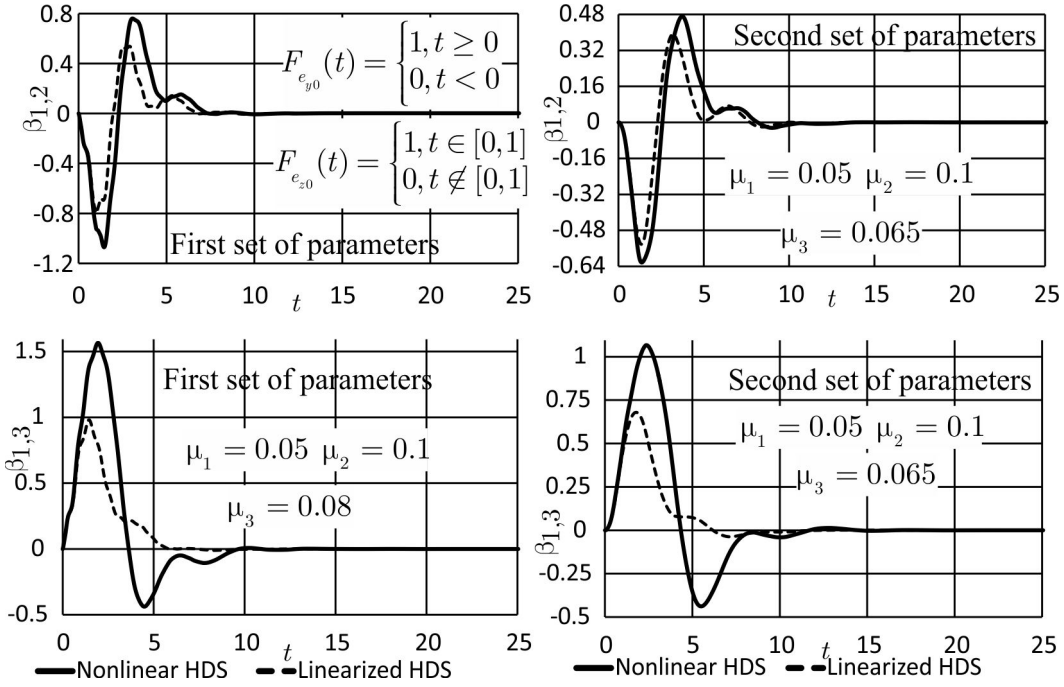

Fig. 3. Output vector functions of linear and nonlinear systems

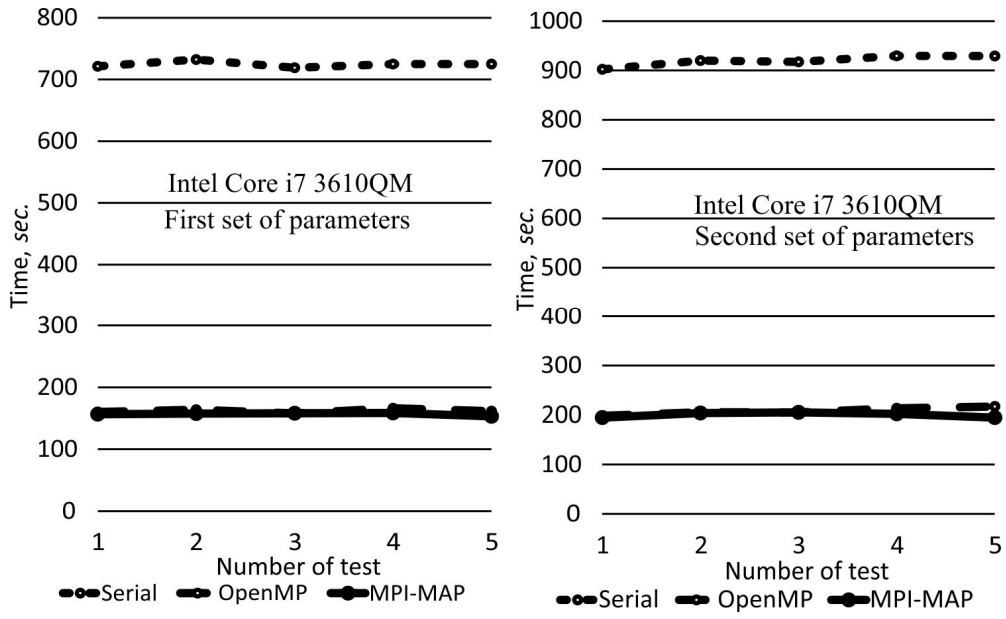

Fig. 4. Comparison of efficiency of MPI-MAP and OpenMP on SMP system

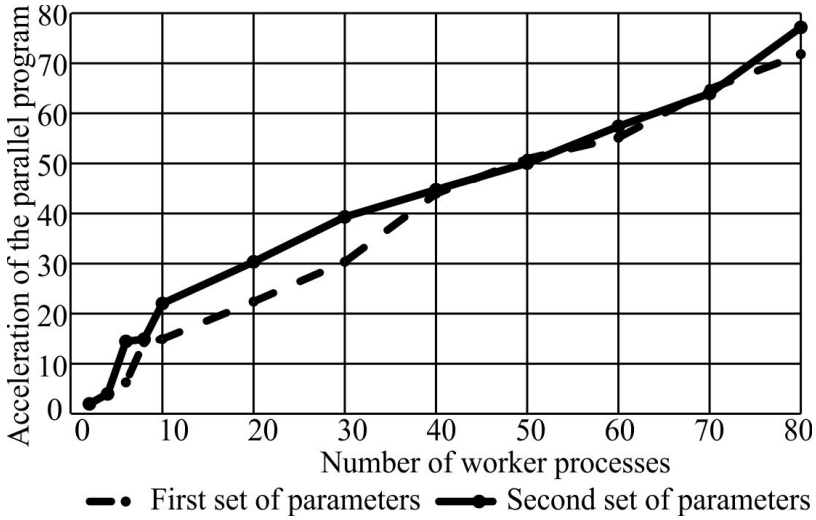

Fig. 5. Computing acceleration depending on the number of the workers

and standard deviations (6) in variation ranges of nonlinearity parameters μ examined above was held using the same spatial grids on the high-performance computing cluster of VRCNIT, Saratov State University. Each computing node of the cluster has two 4-core Intel Xeon E5405 CPUs. As seen from the results represented on fig. 5, on a small number of workers the computational process also accelerates superlinearly by using CPU caches more effectively. Further increasing the number of worker processes gives semi-linear acceleration ratio comparable to the number of workers.

As seen from the results above, using the MPI-MAP pattern is effective when time that program runs in single worker is much more than time needed for data interchange, regardless of the number of workers.

6. Testing the efficiency of parallel computing on Intel Xeon Phi co-processors

Table 1 below shows the representative time of computation of maximum and standard deviations (6) of output vector function of the nonlinear HDS (3) from output vector function of the linearized HDS for the stabilization system using first set of constructive parameters ([6], p. 110) with $\mu_1 \in [0, 0.05]$, $\mu_2 \in [0, 0.1]$ и $\mu_3 \in [0, 0.08]$. Here the spatial grid of parameters μ consisted of 9 nodes for each variable, $t_{max} = 200$. The calculation was held on a node (high-performance computing cluster of CSIT department, Saratov State University) having two 4-core Intel Xeon E5-2603 v2 CPUs

and three Intel Xeon Phi 5110P co-processors (60 cores, 1.053 *GHz* clock rate). Parallel algorithm was implemented using OpenMP technology with default thread mapping as well as using MPI-MAP pattern with dynamic load balancing on 120 worker processes. It is worth noting that running more than 120 worker processes with MPI decreased efficiency. The parallel programs were built for running entirely on Intel Xeon Phi co-processors using –mmic command line key of Intel C++/Fortran compilers.

Table 1

Modeling time for the first set of parameters, *sec.*

Processor, seq./par.	Test 1	Test 2	Test 3	Test 4	Test 5
Intel Xeon E5-2603 v2, seq.	4092	4085	4099	4097	4095
2 x Intel Xeon E5-2603 v2, OpenMP	537	539	537	536	536
Intel Xeon Phi 5110P coprocessor, OpenMP	373	384	376	372	374
Intel Xeon Phi 5110P co-processor, MPI	526	531	536	519	533

Table 2 represents analogous results for calculation time of values (6) for stabilization system using second set of constructive parameters ([7], p. 110) with $\mu_1 \in [0, 0.05]$, $\mu_2 \in [0, 0.1]$, $\mu_3 \in [0, 0.065]$. The spatial grid of parameters $\boldsymbol{\mu}$ also consisted of 9 nodes for each variable, $t_{max} = 200$. In both cases using only one Intel Xeon Phi co-processor with OpenMP threaded model is more effective than using two 4-core Intel Xeon CPUs.

Table 2

Modeling time for the second set of parameters, *sec.*

Processor, seq./par.	Test 1	Test 2	Test 3	Test 4	Test 5
Intel Xeon E5-2603 v2, seq.	5175	5180	5169	5182	5178
2 x Intel Xeon E5-2603 v2, OpenMP	673	677	674	670	673
Intel Xeon Phi 5110P co-processor, OpenMP	448	445	446	452	442
Intel Xeon Phi 5110P co-processor, MPI	653	636	669	642	663

As seen from the results above, one Intel Xeon Phi co-processor is 1.5 times more effective than two 4-core Intel Xeon CPUs and 11 times faster compared to sequential version of the program for tasks with significant parallel potential due to its massive-parallel architecture. At the same time, Intel Xeon phi co-processors have lower clock rate and power consumption. When using Intel Xeon Phi co-processors, parallel execution with OpenMP technology gives the most advantage. Running many MPI processes on a co-processor is less efficient.

It is worth noting that nodes of the modern cluster systems allow using multiple (e.g. three or four) Intel Xeon Phi co-processors at once. This allows to achieve significant acceleration ratio at low power consumption. Data represented in tables 1 and 2 allow supposing that the following strategies are effective on such cluster systems.

The first strategy provides for using mixed parallel program (offloading model) which runs on both CPUs and co-processors. Using OpenMP parallel technology is advisable within host CPUs as well as within co-processors because it provides standard facilities for dynamic balancing of computing load. At the node level, the dynamic load balancing between host CPUs and co-processors can be implemented based on MPI-REDUCE pattern and "manager-workers" strategy similar to MPI-MAP behavior. At the cluster level, the dynamic load balancing can be implemented with MPI-MAP.

Second strategy provides for using co-processors as a virtual node of the cluster running multiple parallel OpenMP threads within single MPI worker process. Single or multiple worker processes with multiple OpenMP threads can be run on CPUs within single node. Dynamic load balancing between multiple MPI processes running on the same or different cluster nodes can be implemented with MPI-MAP pattern.

Conclusion

MPI-MAP pattern as well as standard facilities of OpenMP parallel programming technology successfully solve the problem of dynamic balancing of computing load on symmetric multiprocessor systems with shared memory.

Parallel facilities of thread-based OpenMP technology is preferred within Intel Xeon Phi co-processors.

The advantages of MPI-MAP pattern uncover fully on massive parallel processing clusters with large number of nodes. On such systems, MPI-MAP provides for high acceleration ratio close to the number of worker processes, even on a large scale, for massive parallel tasks. The dynamic load balancing between separate MPI processes running on the same or different cluster nodes can be implemented using MPI-MAP pattern when Xeon Phi co-processors act as virtual nodes.

References (GOST 7.1:2006)

1. Power and Performance Benchmark Methodology 2.1. [Electronic resource] / Standard Performance Evaluation Corporation (SPEC), 2016. – Available to: http://www.spec.org/power/docs/SPEC-Power_and_Performance_Methodology.pdf. – 31.01.2016.

2. Energy-efficient Algorithms for Ultrascale Systems [Electronic resource] / J. Carretero, S. Distefano, D. Petcu, D. Pop, T. Rauber, G. Rünger, D. E. Singh // Supercomputing Frontiers and Innovations, 2015. – Available to: http://www.superfri.org/superfri/article/view/41/35. – 31.01.2016.

3. Andrews, G. R. Foundations of Multithreaded, Parallel and Distributed Programming [Text] / Gregory R. Andrews. – Addison Wesley, 2000. – 664 p.

4. OpenMP Application Program Interface. Version 4.5 - November 2015. [Electronic resource]/ OpenMP Architecture Review Board, 2015. – Available to: http://www.openmp.org/mp-documents/openmp-4.5.pdf. – 31.01.2016.

5. MPI: A Message-Parsing Interface Standard 3.1. June 4, 2015. [Electronic resource]/ Message Passing Interface Forum, 2015. – Available to: http://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf. – 31.01.2016.

6. Andreichenko, D. K. Pattern MPI-MAP and modeling of output vector functions of nonlinear hybrid dynamic systems [Text] / D. K. Andreichenko, K. P. Andreichenko, D. V. Melnichuk // In: Reports of Military Sciences Academy. – 2015. – № 2 (66). – P. 101-115.

7. Concurrency Runtime [Electronic resource]/ Microsoft, 2016. – Available to: http://msdn.microsoft.com/ru-ru/library/dd504870.aspx. – 31.01.2016.

8. Andreichenko, D. K. On the theory of hybrid dynamical systems [Text] / D. K. Andreichenko, K. P. Andreichenko // In: Journal of Computer and Systems Sciences International. – 2000. – Vol. 39, No. 3. – P. 383-398.

9. Andreichenko, D. K. Dynamic analysis and choice of parameters of a model of gyroscopic integrator of linear accelerations with floating platform [Text] / D. K. Andreichenko, K. P. Andreichenko // In: Journal of Computer and Systems Sciences International. – 2008. – Vol. 47, No. 4. – P. 570-583.

10. Andreichenko, D. K. Parameter Selection and Dynamic Analysis of Gas Jet Stabilization Systems with Elastic Rods [Text] / D. K. Andreichenko, K. P. Andreichenko, M. S. Komarova // In: Journal of Computer and Systems Sciences International. – 2012. – Vol. 51, No. 4. – P. 573-586.

11. Andreichenko, D. K. Parallelization of parametric synthesis by "problems portfolio" scheme based on MPI technology [Text] / D. K. Andreichenko, A. A. Eroftiev, D. V. Melnichuk // In: Izv. Saratov Univ. (N. S.), Ser. Math. Mech. Inform. – 2015. – Vol. 15, No. 2. – P. 222-228. DOI: 10.18500/1816-9791-2015-15-2-222-228

12. Hairer, E. Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems [Text] / E. Hairer, G. Wanner. – Springer, 1996. – 614 p.

References (BSI)

1. *Power and Performance Benchmark Methodology 2.1. Standard Performance Evaluation Corporation (SPEC)*, 2016. Available to: http://www.spec.org/power/docs/SPEC-Power_and_Performance_Methodology.pdf (accessed 31.01.2016).

2. Carretero, J., Distefano, S., Petcu, D., Pop, D., Rauber, T., Rünger, G., Singh, D. E. *Energy-efficient Algorithms for Ultrascale Systems. Supercomputing Frontiers and Innovations*, 2015. Available to: http://www.superfri.org/superfri/article/view/41/35 (accessed 31.01.2016).

3. Andrews, G. R. *Foundations of Multithreaded, Parallel and Distributed Programming*. Addison Wesley, 2000. 664 p.

4. *OpenMP Application Program Interface. Version 4.5*. November 2015. OpenMP Architecture Review Board, 2015. Available to: http://www.openmp.org/mp-documents/openmp-4.5.pdf (accessed 31.01.2016).

5. *MPI: A Message-Parsing Interface Standard 3.1. June 4, 2015. Message Passing Interface Forum*, 2015. Available to: http://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf (accessed 31.01.2016).

6. Andreichenko, D.K. Andreichenko, K.P., Melnichuk, D.V. *Pattern MPI-MAP i modelirovanie vykhodnykh vektor-funktsii nelineinykh kombinirovannykh dinamicheskikh sistem* [Pattern MPI-MAP and modeling of output vector functions of nonlinear hybrid dynamic systems]. *Doklady Akademii voennykh nauk – Reports of Military Sciences Academy*, 2015, no. 2 (66), pp. 101-115.

7. *Concurrency Runtime.* Microsoft, 2016. Available to: http://msdn.microsoft.com/ru-ru/library/dd504870.aspx. (accessed 31.01.2016).

8. Andreichenko, D. K., Andreichenko, K. P. On the theory of hybrid dynamical systems. *Journal of Computer and Systems Sciences International*, 2000, vol. 39, no. 3, pp. 383-398.

9. Andreichenko, D. K., Andreichenko, K. P. Dynamic analysis and choice of parameters of a model of gyroscopic integrator of linear accelerations with floating platform. *Journal of Computer and Systems Sciences International*, 2008, vol. 47, no. 4, pp. 570-583.

10. Andreichenko, D. K., Andreichenko, K. P., Komarova, M. S. Parameter Selection and Dynamic Analysis of Gas Jet Stabilization Systems with Elastic Rods. *Journal of Computer and Systems Sciences International*, 2012, vol. 51, no. 4, pp. 573-586.

11. Andreichenko, D. K., Eroftiev, A. A., Melnichuk, D. V. Parallelization of parametric synthesis by "problems portfolio" scheme based on MPI technology. *Izv. Saratov Univ. (N. S.), Ser. Math. Mech. Inform,* 2015, vol. 15, no. 2, pp. 222-228. DOI: 10.18500/1816-9791-2015-15-2-222-228

12. Hairer, E., Wanner, G. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer, 1996. 614 p.

Поступила в редакцию 3.02.2016, рассмотрена на редколлегии 14.04.2016

ДИНАМІЧНЕ БАЛАНСУВАННЯ ОБЧИСЛЮВАЛЬНОГО НАВАНТАЖЕННЯ ПРИ ВИРІШЕННІ ЗАВДАНЬ З ВИСОКИМ СТУПЕНЕМ ПАРАЛЕЛІЗМУ

Д. К. Андрейченко, Д. В. Мельнічук, А. О. Єрофтієв

Оптимізація алгоритмів для паралельних обчислювальних систем вимагає ефективної динамічного балансування обчислювального навантаження. Технології OpenMP і MPI дозволяють розробляти програмне забезпечення для більшості сучасних паралельних обчислювальних архітектур, проте MPI не передбачає стандартних засобів для динамічного балансування обчислювального навантаження. Для вирішення зазначеної проблеми запропонований патерн розпаралелювання MPI-MAP, який реалізує стандартними засобами MPI етап MAP відомого патерну MAP-REDUCE. Ефективність динамічного балансування обчислювального навантаження на основі OpenMP і MPI-MAP показана для симетричних мультипроцесорних систем зі спільною пам'яттю, кластерних систем з розподіленою пам'яттю і обчислювальних систем з сопроцесорами-прискорювачами Intel Xeon Phi. Як приклад розглянуто задачу чисельного моделювання впливу типових нелінійностей на вихідні вектор-функції нелінійної системи стабілізації рухомого об'єкту управління.

Ключові слова: MPI, OpenMP, паралельне програмування, комбіновані динамічні системи.

ДИНАМИЧЕСКАЯ БАЛАНСИРОВКА ВЫЧИСЛИТЕЛЬНОЙ НАГРУЗКИ ПРИ РЕШЕНИИ ЗАДАЧ С ВЫСОКОЙ СТЕПЕНЬЮ ПАРАЛЛЕЛИЗМА

Д. К. Андрейченко, Д. В. Мельничук, А. А. Ерофтиев

Оптимизация алгоритмов для параллельных вычислительных систем требует эффективной динамической балансировки вычислительной нагрузки. Технологии OpenMP и MPI позволяют разрабатывать программное обеспечение для большинства современных параллельных вычислительных архитектур, однако MPI не предусматривает стандартных средств для динамической балансировки вычислительной нагрузки. Для решения указанной проблемы предложен паттерн распараллеливания MPI-MAP, реализующий стандартными средствами MPI этап MAP известного паттерна MAP-REDUCE. Эффективность динамической балансировки вычислительной нагрузки на основе OpenMP и MPI-MAP показана для симметричных мультипроцессорных систем с общей памятью, кластерных систем с распределенной памятью и вычислительных систем с сопроцессорами-ускорителями Intel Xeon Phi. В качестве примера рассмотрена задача численного моделирования влияния типовых нелинейностей на выходные вектор-функции нелинейной системы стабилизации подвижного объекта управления.

Ключевые слова: MPI, OpenMP, параллельное программирование, комбинированные динамические системы

Андрейченко Дмитрий Константинович – д-р физ.-мат. наук, зав. кафедрой математического обеспечения вычислительных комплексов и информационных систем, ФГБОУ ВО «Саратовский национальный исследовательский государственный университет имени Н. Г. Чернышевского», Саратов, Россия, e-mail: kp_andreichenko@renet.ru.

Мельничук Дмитрий Вадимович – магистрант факультета компьютерных наук и информационных технологий, ФГБОУ ВО «Саратовский национальный исследовательский государственный университет имени Н.Г. Чернышевского», Саратов, Россия, e-mail: meldm007@gmail.com.

Ерофтиев Андрей Александрович – зам. начальника отдела информационных ресурсов и систем Поволжского центра новых информационных технологий, ФГБОУ ВО «Саратовский национальный исследовательский государственный университет имени Н. Г. Чернышевского», Саратов, Россия, e-mail: eroftiev.andrey@gmail.com.

Andreichenko Dmitry Konstantinovich – Doctor of Physical and Mathematical Sciences, Head of Department of Mathematical Software for Computer and Information Systems, Saratov State University, Saratov, Russia, e-mail: kp_andreichenko@renet.ru.

Melnichuk Dmitry Vadimovich – master student of Department of Computer Science and Information Technology, Saratov State University, Saratov, Russia, e-mail: meldm007@gmail.com.

Eroftiev Andrey Aleksandrovich – Deputy Head of Department of Information Resources and Systems of Volga Regional Center of New Information Technologies, Saratov State University, Saratov, Russia, e-mail: eroftiev.andrey@gmail.com.