HIGH PERFORMANCE CLUSTER COMPUTING ON A FLYING SIMULATOR

Peter Kvasnica[1], Igor Kvasnica[2],

[1]Alexander Dubček University of Trenčín, Center of Information Technology,
911 50 Trenčín, Študentská 2, Slovak Republic
[2]Kardinal, spol. s r.o.,
911 05 Trenčín, Kvetná 25, Slovak Republic

kvasnica@tnuni.sk (Peter Kvasnica)

Abstract

The article covers a design of parallel computing in information systems of a simulator. The concept is based on computers that create a distributed computer system of a flight simulator. This information system is created by computers and program applications of mathematic models. An important part of this article describes high performance computing with tasks, a compute cluster, a job scheduler and parallel execution. It explains job admission by a command line interface and creation of these jobs, mathematic models of a flying simulator. Mathematic modeling is the art of transformation of a problem from an original application into a theoretic area to mathematical formulations for a numerical analysis.
A significant part of this article describes the implementation of aircraft computation speed depending on fuel supply and an elevator, high performance computing implemented by single-processor architecture. Simulation of a horizontal and vertical flight is defined by the angles and angular velocities around axises of the aircraft. This is accomplished by some computers, which are able to create a distributed computer system for a flight simulator. The programme loop for calculation parameters of mathematic model is created on the high performance cluster. This distributed mathematical model of an aircraft in longitudinal direction speed computes in real time. Flying simulator modeling processes on these computers also create a time benefit in a parallel system.

Keywords: high performance computing, cluster, parallel task, mathematic model, flying simulator.

Authors' Biographies

Peter Kvasnica. He has been spending several years by researching mathematical models of flying objects and programming virtual reality applications. He publishes in the area of application of mathematical methods of flying objects, in scientific programs with the emphasis on modelling of such systems. He graduated in Technical University of Brno (VUT Brno), and was awarded by Doctor of Philosophy (PhD.) of M. R. Štefánik Military Academy of Aviation in Košice. He is involved in the development of adapted mathematical models and use of distributed computer system in flight simulators for real-time applications.

1 Introduction

Future high - performance computing (HPC) systems, in which users have access to application and system services, will need support in a traditional batch execution system. As the use of high performance spreads to various application domains, some services will rely on immediate and interactive program execution. These characteristics will need to reserve resources, while some others will need a varying set of processors. Hardware and software applications of a defined computer system compute by a mathematic model of a flying simulator in real time.

2 An Overview of a Compute Cluster Server for Operations

User regularly prepares a job to run in a compute cluster, the job runs through three stages (admission, allocation and activation).

Microsoft® Windows® Compute Cluster Server 2003 brings high-performance computing (HPC) to industry standard, low-cost servers. *Jobs* – discrete activities scheduled to perform on a compute cluster. In some situations, tasks are serial – running one after another; in others, they are parallel – running all at the same time [1].

A basic principle of job operation in Windows Compute Cluster Server 2003 relies on three key concepts:

1. Admission, or job submission;
2. Allocation, or job scheduling;
3. Activation or job launch.

These three concepts form an underlying structure of a job life cycle in HPC. Fig. 1 illustrates a relationship between each aspect of job operation. Every time a user prepares a job to run in a compute cluster, the job runs through the three stages.

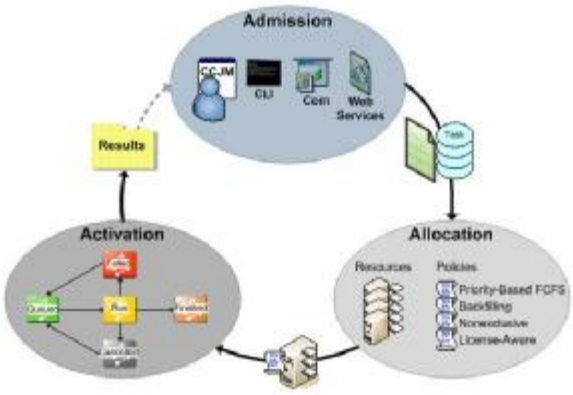


Fig. 1 The HPC job life cycle

The cluster itself consists of a *head node* and *compute nodes.* The head node is designed to run by a job scheduler, add or remove compute nodes, view job and node status. In other words, the head node manages cluster operations [1]. Compute nodes are designed to run application jobs.

A *cluster* is a top-level organizational unit of an HPC cluster platform. A cluster consists of the following elements [2]:

1. *Node* – a single compute node with one or more processors;
2. *Queue* – an organizational unit that provides queuing and job scheduling;
3. *Job* – a collection of tasks that a user initiates.

A *task* represents execution of a program on given compute nodes. A task can be a serial program (single process) or a Message Passing Interface (MPI) program with multiple parallel processes.

The *Job Scheduler* queues jobs and their sub-tasks. It allocates resources to these jobs; initiates tasks on compute nodes of the cluster, and monitors a status of jobs, tasks, and compute nodes. Job scheduling is performed through a set of rules called *scheduling policies.*

3 Parallel Execution Support

Users would like to use various programming languages that suit their needs and personal preferences while enjoying platform independence and reliable execution, the use of a particular programming language, execution mode, and the like [3].

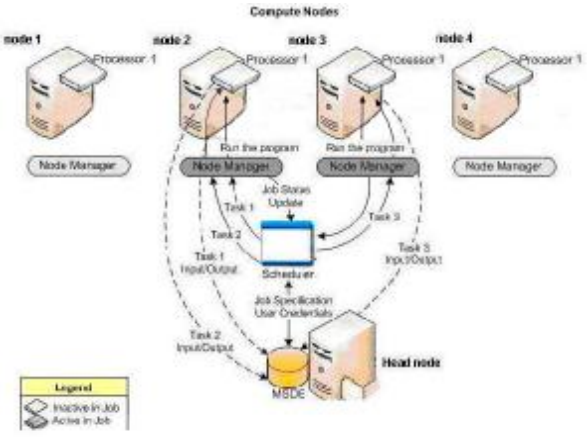


Fig. 2 Parallel task execution on HPC

Tasks operate in either serial or parallel mode. In a *serial mode,* tasks run sequentially on available resources in the nodes. Fig. 2 illustrates how a task 1 is assigned to a processor on the second node; then a task 2 is assigned to the same processor, a task 3 moves to a processor of a third node, and so on.

Parallel tasks typically call upon Microsoft® Message Passing Interface (MPI) software (called *MS MPI*)

through the mpiexec task launcher on compute nodes [4]. Task processes are started through the MPI-specific nodal daemon for Single Program Multiple Data (SPMD) programming.

Users create jobs by first specifying the job properties, including priority, run time limit, number of required processors, requested nodes, and a node exclusivity. After defining the job properties, users can assign tasks to the job. Users use either Compute Cluster Job Manager or CLI to create jobs.

4 Development Issues and Compiling

First, build a Windows XP or Windows Server 2003 machine. Place this machine in an MPICH domain and create a user in the domain with the same name as your MCS user name. Use your MCS password for this new account. This isn't a requirement; you can create any username you want with any password you want and it doesn't have to be part of the MPICH domain [8].

To build MPICH2, you will need:

1. Microsoft Developer Studio .NET 2003;
2. Microsoft Platform SDK;
3. Cygwin - full installation;
4. Intel Fortran compiler IA32;
5. Intel Fortran compiler EMT64.

The easiest way to build an MPICH2 distribution is to use a Developer Studio environment and a makewindist.bat script from the top level of an mpich2 source tree. You can check out mpich2 from CVS or you can simply copy this batch file from the distribution. The batch file knows how to check out mpich2 so it is the only file required to make a distribution. The product GUIDs needs to be changed when a new release is created.

The makefile in an mpich2\winbuild directory builds a distribution based on what compilers are specified in the environment so it can be used to compile any version of MPICH2. The following targets can all be built with this mechanism:

- Win64 X64
- Win64 IA64
- Win32 x86

But first you need to have mpich2 checked out and configured before building.

This section describes how to set up a project to compile an MPICH2 application MS Developer Studio NET 2003.

1. Create a project and add your source files.
2. Navigate to Configuration Properties::C/C++::General
3. Add C:\Program Files\MPICH2\include to the "Additional Include Directories" box.
4. Navigate to Configuration Properties::Linker::General
5. Add C:\Program Files\MPICH2\lib to the "Aditional Library Directories" box.
6. Navigate to Configuration Properties::Linker::Input
7. Add cxx.lib and mpi.lib and fmpich2.lib to the "Additional Dependencies" box. If your application is a C application then it only needs mpi.lib [8].

4.1 User Credentials

User credentials mpiexec must have the user name and password to launch MPI applications in the context of that user. Run mpiexec - register to save your username and password. Then mpiexec will not prompt you for this information.

The user context under which the script is run must have credentials saved so mpiexec doesn't prompt for them. So scripts won't hang, mpiexec provides a flag, -noprompt, that will cause mpiexec to print out errors in cases when it normally would prompt for a user input. This can also be specified in the environment with the variable MPIEXEC NOPROMPT.

You can also save more than one set of user credentials. Add the option -user n to the -register, -remove, -validate, and mpiexec commands to specify a saved user credential other than the default. The parameter n is a non-zero positive number.

For example this will save credentials in slot 1:

mpiexec -register -user 1.

And this command will use the user 3 to launch a job:

mpiexec -user 3 -n 4 mathmod.exe.

4.2 MPICH2

MPICH2 for Windows comes with multiple complete implementations of MPI. These are called channels and each build represents a different transport mechanism used to move MPI messages. The default channel uses sockets for communication. There is a channel that uses both sockets and a shared memory [8]. There is a channel that uses Infiniband. And has a thread-safe version of a sockets channel. Short names for the channels are: sock, shm, sshm, ssm, mt.
These channels can be selected at runtime with an environment variable: MPICH2 CHANNEL. The following is an example that uses an Infiniband channel instead of a default sockets channel:

mpiexec -env MPICH2_CHANNEL ib -n 4 math.exe

or

mpiexec -channel ib -n 4 math.exe.

Windows comes with a default firewall that is usually turned on by default. Firewalls block all TCP ports by default which renders MPICH2 applications inoperable because sockets on arbitrary ports assigned

by the operating system are a default communication mechanism used by MPICH2.

5 Mathematic Modeling in an MPI Flying Simulator

Mathematic model of a flying simulator is running at a cluster computer system, Fig. 2. Mathematical modeling is the art of translating problems from an application area into tractable mathematical formulations. Theoretical and numerical analyses are provided inside the system and are fully available to users. They are generally important for accurate modeling.

From [5] mathematic model in transfer function for longitudinal direction speed arise ΔV(s) (Δ = increase in value), with two input values $\Delta\delta_T(s)$, $\Delta\ \delta_B$ (s) is defined:

$$\Delta V(s) = -W_V^{d_T}(s)\Delta d_T(s) - W_V^{d_B}(s)\Delta d_B(s), \tag{1}$$

where

$$W_V^{d_T}(s) = -a_x^{dT}\frac{\Delta_{11}(s)}{\Delta(s)},$$

$$W_V^{d_B}(s) = -a_y^{dB}\frac{\Delta_{21}(s)}{\Delta(s)} - a_{mz}^{dB}\frac{\Delta_{31}(s)}{\Delta(s)},$$

s – Laplace operator in differential equations, Δ(s) – determinant of a transfer function, $\Delta_{ij}(s)$ – minor of the determinant a given element a_{ij} (i-th row and j-th column) in system equations. Coefficients of a transfer function are computed next [5]:

$$a_x^{dT} = \frac{-\partial\Delta P}{\partial x} = -5,$$

$$a_y^{dB} = \frac{-\partial\Delta Y}{\partial dB} = -0{,}11$$

$$a_{mz}^{dB} = \frac{-\partial\Delta mz}{\partial dB} = -0{,}42 \tag{2}$$

In a numeric formulation, the transfer function amount increase speed of fuel supply Eq. (1) has a form:

$$W_V^{d_T}(s) = 5\frac{s^3+1{,}12s^2+62{,}78s+25{,}32}{s^4+1{,}13s^3+62{,}8s^2+28{,}66s+4{,}09}\Delta d_T(s). \tag{3}$$

In a numeric formulation, the transfer function speed of elevator Eq. (1) has a form:

$$W_V^{d_B}(s) = \frac{-0{,}11(9{,}81s+620{,}97)-0.42(-9{,}81s-10{,}01)}{s^4+1{,}13s^3+62{,}8s^2+28{,}66s+4{,}09}\Delta d_B(s). \tag{4}$$

From [5] mathematic model in transfer function for longitudinal direction angle of attack arise Δα(s) (Δ = increase in value), with two input values $\Delta\delta_T(s)$, $\Delta\ \delta_B$ (s) is defined:

$$\Delta a(s) = -W_a^{d_T}(s)\Delta d_T(s) - W_a^{d_B}(s)\Delta d_B(s), \tag{5}$$

where

$$W_a^{d_T}(s) = -a_x^{dT}\frac{\Delta_{12}(s)}{\Delta(s)},$$

$$W_a^{d_B}(s) = -a_y^{dB}\frac{\Delta_{22}(s)}{\Delta(s)} - a_{mz}^{dB}\frac{\Delta_{32}(s)}{\Delta(s)},$$

s – Laplace operator in differential equations, Δ(s) – determinant of a transfer function, $\Delta_{ij}(s)$ – minor of the determinant a given element a_{ij} (i-th row and j-th column) in system equations [7].
In a numeric formulation, the transfer function amount increase speed on fuel supply Eq. (5) has a form:

$$W_a^{d_T}(s) = 5\frac{0{,}002\,s^2 - 0{,}2518\,s - 0{,}1}{s^4+1{,}13s^3+62{,}8s^2+28{,}66s+4{,}09} * \Delta d_T(s). \tag{6}$$

In a numeric formulation, the transfer function speed of elevator Eq. (5) has a form:

$$W_a^{d_B}(s) = \frac{-0{,}11(-s^3+0{,}8862s^2+0{,}012422s-2{,}4525)}{s^4+1{,}13s^3+62{,}8s^2+28{,}66s+4{,}09} \frac{-0.42(-s^2-0{,}4138s-0{,}02514)}{s^4+1{,}13s^3+62{,}8s^2+28{,}66s+4{,}09}\Delta d_B(s). \tag{7}$$

In Eq. (3), (4) or (6), (7), respectively the unit step input $\Delta\delta_T(s)$, i. e. $\Delta\delta_T(s)=1/s$ or $\Delta\delta_B(s)$, respectively, i. e. $\Delta\delta_B(s)=1/s$.

6 Models of a Flying Simulator in a Cluster

Mathematic modeling at a computer system is realized by 5 computers (1 head node and 4 compute nodes). Each computer has a Pentium IV 2.4 GHz processor, memory 512 MB RAM, hard disk 40 GB 5400 rpm, fast Ethernet card 100 Mb/s. The head node computer has same parameters. The Scheduler runs on a stand alone computer (a head node computer) connected to a real computer system through Ethernet.

This section describes important options of mpiexec.exe

- -n x or -np x specify the number of processes to launch. In our case – n 5.
- -localonly x or –localonly, specify that the processes should only be launched on a local host. This option can replace the -n x option or be used in conjunction with it when it is only a flag.
- -machinefile filename. Use the specified file to get host names to launch processes.

• -hosts n host1 host2 host3 ... Specify that the processes should be launched on a list of hosts. This option replaces the -n x option.
• -hosts n host1 m1 host2 m2 host3 m3 ... Specify that the processes should be launched on a list of hosts and how many processes should be launched on each host. A total number of processes launched is m1 + m2 + m3 + ... mn. In our case, n 193.87.64.122, 193.87.64.112, 193.87.64.110, 193.87.64.101, 193.87.64.120.

To start a job through the CLI, type the following command **mpiexec.exe**, see Fig. 3. There is a desktop wrapper in the figure. It normally would prompt the user to input services. A radio button defines the path name of application mathematic models "empichtest.exe". A user defined number of processors is 5 (host 5), see Fig. 3. The number of IP addresses of computers that create an MPI compute cluster (hosts) is defined in the host dialog box. Short names for the channel rewrite sock, nonblocked communication between computers.

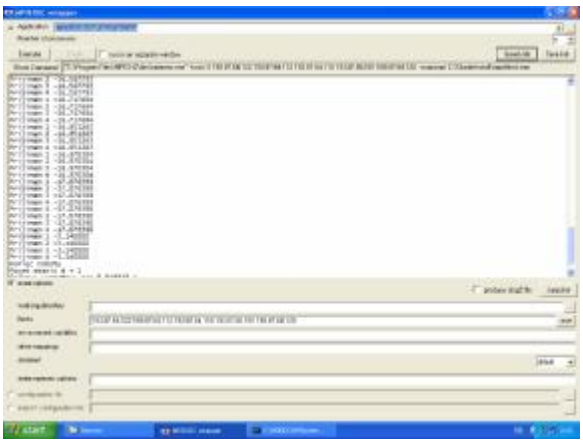


Fig. 3 Dialog window mathematic models of a flying simulator in MPI

Node1 execution file, that represents an execution program (an aircraft job) of the part of a mathematic model of aircraft speed dependence on fuel supply Eq. (3). *Node2* execution program of the part of a mathematic model of aircraft speed dependence on the elevator Eq. (4). *Node3* represents the execution program of the part of a mathematic model of aircraft attack angle dependence on fuel supply Eq. (6). *Node4* execution program of the part of a mathematic model of aircraft attack angle dependence on the elevator Eq. (7). Source code of mathematic models of aircraft is written in C++ language. This approache provides much flexibility for confirmation of the job of aircraft in cluster [6].

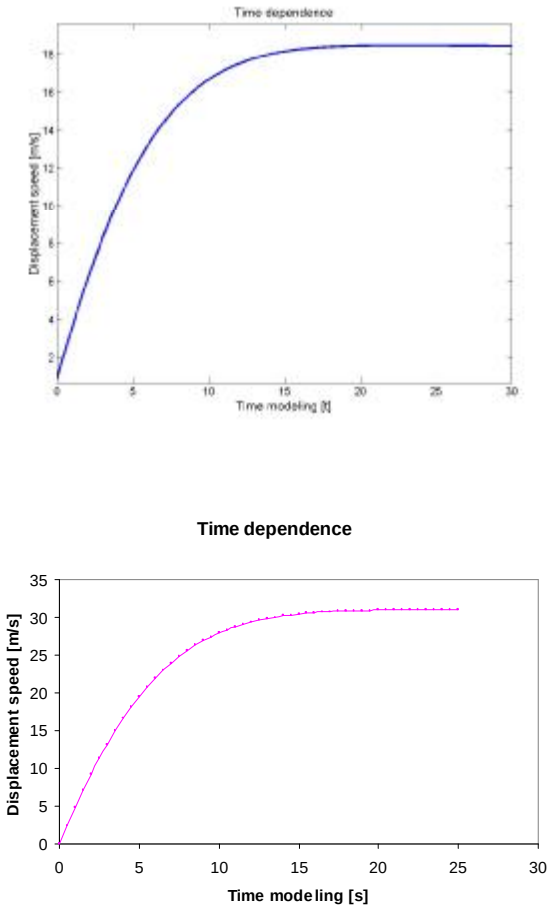
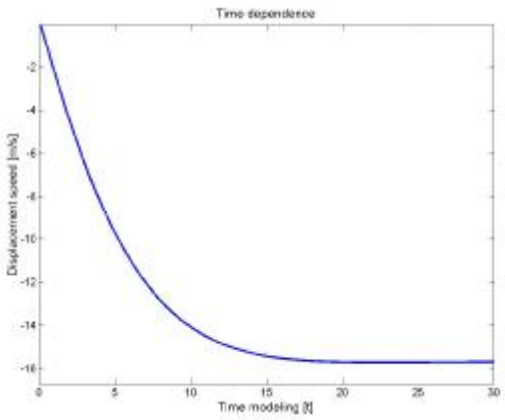


Fig. 4 System response Eq. (3) to a unit step input $\Delta\delta_T(s)$, in Matlab (up), model of at the flying simulator in MPI - node 1 (down)

The Fig. 4 compares results of a compute time response to the unit step of a mathematic model in an increment speed of aircraft Eq. (3) in the Matlab or in the node 1 the MPI system, respectively. From comparison of results it can be derived that the speed change dynamics is approximately the same in both Matlab and MPI at the node 1. The steady state of a speed increment is in both systems compatible.



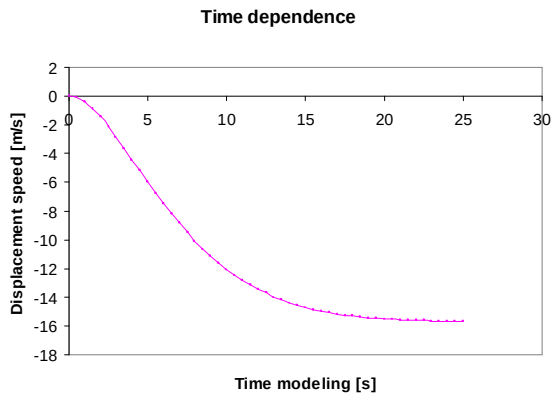


Fig. 5 System response to a unit step input $\Delta\delta_B(s)$, in Matlab (up), model of at the flying simulator in MPI - node 2 (down)

The Fig. 5 compares a graphics output compute time of a response to the unit step a mathematic model in increment speed of aircraft Eq. (4) in the Matlab or in the node 2 the MPI system, respectively. From comparison of results it can be derived that the speed change dynamics is approximately the same in both Matlab and MPI at the node 2. The steady state of a speed increment is in both systems compatible.

The results from speed modeling on high performance computing are shown in the table below.

Tab. 1 Steady state of a mathematic models on a stand alone computer versus cluster computers (MPI)

Mathematic model	**Single computer Matlab**		**Cluster, MPI**		
	Steady state	**Time [s]**	**Steady state**	**Time [s]**	**Distort [%]**
Speed from fuel supply	31.1 m/s	22-23	31.01 node 1	24	-0.3
Speed from elevator	-15.61 m/s	20-21	-15.68 node 2	23	0,5
Angle of attack on fuel supply	7.03 deg	21-22	7.02 deg node 3	20	-0.2
Angle of attack on elevator	-3.93 deg	20-21	-3.94 deg node 4	22	0.3

7 Conclusion and Future Work

This paper described our approach to support of a computational application in a dynamic mathematic model of a flying simulator by high - performance computing.

Father tests and evaluations are being conducted continuously to determine reliability of our implementations and to determine performance and overheads of the system, respectively.

Opportunities for future development include:

- Real-time aircraft mathematic model implementation
- Aircraft mathematic model multitasking processes
- Web-based computing.

The system demonstrated that with these application processes of mathematic models programmers can create highly adaptable, dynamic, service-oriented applications for a flying simulator.

8 References and Citations

1 Nelson Ruest and Danielle Ruest: Using Microsoft Windows Compute Cluster Server 2003 Job Scheduler, November 2005.

2 Pota, S., Sipos, G., Juhasz, Z., Kacsuk, P.: Distributed and Parallel Systems, Parallel Program Execution Support in the JGRID System, Springer Science, New York 2005, ISBN 0-387-23094-7, p. 13-19.

3 Feitelson, D., G., Rudolf, L.: "Parallel Job Scheduling: Issues and Approaches" Lecture Notes in Computer Science, Vol. 949, p. 1-xx, 1995.

4 Sun Microsystems, Jini Technology Core Platform Specification, http://www.sun.com/jini/specs, 2005.

5 Krasovskij, A., A.: *Sistemy avtomaticeskogo upravlenja poletom i ich analiticeskoje konstruirovanie*, Nauka, Moskva 1980, pp. 589.

6 John H. Blakelock: Automatic Control of Aircraft and Missiles, John Wiley & Sons. Inc., New York, 1991.

7 Rolfe J.M., Staples K.J.: Flight Simulation, Cambridge University Press, 1986.

8 Ashton D.: MPICH2 Windows Development Guide, Version 1.0.3, Mathematics and Computer Science Division Agronne National Laboratory, 2005