ARTICLE IN PRESS

Available online at www.sciencedirect.com

ScienceDirect

J. Parallel Distrib. Comput. ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮

Journal of Parallel and Distributed Computing

www.elsevier.com/locate/jpdc

A Grid-based Virtual Reactor: Parallel performance and adaptive load balancing

Vladimir V. Korkhov[a,b,*], Valeria V. Krzhizhanovskaya[a,b], P.M.A. Sloot[a]

[a] *University of Amsterdam, Section Computational Science, Netherlands*
[b] *St. Petersburg State Polytechnic University, Russia*

Received 10 June 2006; received in revised form 16 August 2007; accepted 31 August 2007

Abstract

We address the problem of porting parallel distributed applications from static homogeneous cluster environments to dynamic heterogeneous Grid resources. We introduce a generic technique for adaptive load balancing of parallel applications on heterogeneous resources and evaluate it using a case study application: a Virtual Reactor for simulation of plasma chemical vapour deposition. This application has a modular architecture with a number of loosely coupled components suitable for distribution over the Grid. It requires large parameter space exploration that allows using Grid resources for high-throughput computing. The Virtual Reactor contains a number of parallel solvers originally designed for homogeneous computer clusters that needed adaptation to the heterogeneity of the Grid. In this paper we study the performance of one of the parallel solvers, apply the technique developed for adaptive load balancing, evaluate the efficiency of this approach and outline an automated procedure for optimal utilization of heterogeneous Grid resources for high-performance parallel computing.

Keywords: Grid; Adaptive load balancing; Heterogeneous resources; Benchmarking; Virtual Reactor; PECVD

1. Introduction

Porting complex distributed applications to Grid poses a challenge to computer and computational sciences, mostly due to the dynamical and decentralized nature of the Grid. Involving *parallel* computational solvers further complicates the problem because of a severe heterogeneity of Grid resources characterized by a wide range of processing power and network links bandwidth. The scientific community has been investing a lots of effort into the development of Grid-aware problem-solving environments (PSE) for complex applications [1,26]. The importance of fully integrated simulators is recognized by various research groups and scientific software companies [2]. The Virtual Reactor used here as a test case was developed for simulation of plasma enhanced chemical vapour deposition (PECVD), a multiphysics process spanning a wide range of spatial and temporal scales [15,16]. Simulation of three-dimensional flow with chemical reactions and plasma discharge in complex geometries is one of the most resource-demanding problems in computational science, requiring both high-performance and high-throughput computing.

Grid computing technologies opened up new opportunities to access virtually unlimited computational resources, and inspired many researchers to develop new methodologies and algorithms for parallel distributed applications on the Grid. The PECVD Virtual Reactor discussed in this paper has also been on its way to the Grid [15]. It serves as a test-case driving and validating the development of the Russian–Dutch Grid (RDG) computational for distributed high performance simulation [14,22]. The Virtual Reactor is particularly suitable for porting to the Grid since it can be decomposed into a number of functional components (services). In addition to that, this application requires large parameter space exploration, which can be efficiently organized on the Grid. Tools to support distributed parametric modelling on the Grid are being developed,

* Corresponding author. University of Amsterdam, Section Computational Science, Netherlands.
E-mail addresses: vkorkhov@science.uva.nl (V.V. Korkhov), valeria@science.uva.nl (V.V. Krzhizhanovskaya), sloot@science.uva.nl (P.M.A. Sloot).

0743-7315/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2007.08.010

in particular the Nimrod-G middleware [20] which is used in this project. Current work on porting the Virtual Reactor to the Grid started within the framework of the CrossGrid EU project [26] and the Virtual Laboratory for e-Science [27]. Some results of these efforts were reported in [15]. The RDG Grid is the successor of the CrossGrid in a sense that it uses many of the CrossGrid infrastructure services and operates as a testbed for the Virtual Reactor application. The final Grid-based Virtual Reactor PSE aims at being a collaborative system, a distributed scientific workbench with advanced interaction and visualization facilities.

In this paper we address the issue of porting an existing complex PSE from homogeneous cluster environment to dynamic heterogeneous Grid resources. The RDG provides a strong hardware background for this research as it contains sites with both homogeneous and heterogeneous computing and networking resources. To build a Grid-enabled PSE based on a modular application, a proper functional decomposition of modules is required. To assure that the components—especially computational modules—are distributed efficiently, it is necessary to evaluate their performance and behaviour, tracking the dependencies on the input data and computational parameters, pinpointing the scalability and evaluating the influence of the infrastructure parameters.

A countless number of parallel applications have been developed for traditional (i.e. static homogeneous) parallel systems. The real problem in porting such applications to Grid environments is to keep up a high level of parallel efficiency. To assure efficient utilization of Grid resources, special methods for workload distribution control should be applied. Proper workload optimization methods should take into account two aspects: (1) the application characteristics (e.g. the amount of data transferred between the processes, amount of floating point operations and memory consumption) and (2) the resource characteristics (e.g. processors, network and memory capacities, as well as the level of heterogeneity of the dynamically assigned resources). The method should be computationally inexpensive so as to not impose too high overheads. In this paper we present such a method and validate it using one of the parallel solvers of the Virtual Reactor.

The issue of load balancing in a Grid environment is addressed by a number of research groups. A lot of studies on load balancing consider distribution of processes to computational resources on the system/library level with no modifications in the application code [3,9]. Less often, load-balancing code is included into the application source code to improve performance in specific cases [21,23]. Some research projects concern load-balancing techniques that use source code transformations to improve the execution of the application [5]. We employ an application-centric approach where the balancing decisions are taken by the application itself. The algorithm that estimates the resources and suggests the optimal load balancing of a parallel job is generic and can be employed in any parallel application to be executed on heterogeneous resources.

A detailed description of global load optimization approaches for heterogeneous resources and adaptive mesh refinement applications is given in [18,19,29]. However, in [19,29] no network links heterogeneity was considered and only static resource estimation (initialization) was performed in [19,18]. These two issues are the major challenges of Grid computing: (1) the heterogeneity of the network links can be an order of magnitude higher that of the processing power; and (2) Grid resources are inherently dynamic. While developing our algorithm, we tried to facet specifically these two cornerstones. The approaches discussed in [19,29] are only valid for batch sequential applications (specifically for the queuing systems and computer cluster schedulers), whereas our effort is directed towards parallel programs utilizing heterogeneous resources.

The paper is organized as follows: Section 2 describes the algorithm for adaptive workload balancing (AWLB) on heterogeneous resources. Section 3 outlines the architecture of the Virtual Reactor application and the RDG testbed infrastructure. Section 4 presents the results of testing one of the parallel solvers on the RDG homogeneous sites. Section 5 shows the results of applying the load-balancing technique to our case study application. Section 6 draws conclusions and presents directions for future research.

2. Adaptive load balancing on heterogeneous resources

One of the factors that determine the performance of parallel applications on heterogeneous resources is the quality of the workload distribution, e.g. through functional decomposition or domain decomposition. Optimal load distribution is characterized by two things: (1) all processors have a workload proportional to their computational capacity and (2) communications between the processors are minimized. These goals are conflicting since the communication is minimized when all the workload is processed by a single processor and no communication takes place, and distributing the workload inevitably incurs communication overheads. Thus, it is necessary to find a balance and define a metric that characterizes the quality of workload distribution for a parallel problem. One of the existing methods is to introduce a cost function reflecting the application execution time. Minimization of this function corresponds to minimization of the application runtime. The function should be simple and independent of the details of the code. A generic form of the cost function is [6–8]:

$$H = H_{\text{calc}} + \beta H_{\text{comm}}. \tag{1}$$

H_{calc} is minimized when the workload distribution among the processors is proportional to the processors capacity (or equal in case of homogeneous processors), H_{comm} is minimized when the communication time is minimal and β is a parameter that can be varied in order to tune the balance between the calculation and communication terms. This parameter is dependent on both the application requirements and the resources characteristics.

The main generic parameters that influence a parallel application performance are:

- The application parameter $f_c \sim N_{\text{comm}}/N_{\text{calc}}$, where N_{comm} is the total amount of application communications, i.e. data to be exchanged (measured in bit) and N_{calc} is the total

Please cite this article as: V.V. Korkhov, et al., A Grid-based Virtual Reactor: Parallel performance and adaptive load balancing, J. Parallel Distrib. Comput. (2007), doi: 10.1016/j.jpdc.2007.08.010

amount of computations to be performed (measured in Flop);
- The resource parameter $\mu \sim t_{\text{comm}}/t_{\text{calc}}$, where t_{comm} is the typical time taken to communicate a single word between the processors and t_{calc} is the time required to perform a generic floating point calculation.

The product of these two parameters $f_c\mu$ is often called the fractional communication overhead [8].

The goal of load balancing is to minimize the cost function (1). Parameter β in this expression is an aggregated value related to the application and resource-specific parameters f_c and μ. Knowledge of these application and resource properties allows constructing an appropriate form of parameter β to perform suboptimal load distribution [7]. Unfortunately, in most real-life complex simulation problems, it is not possible to theoretically calculate the application specific parameter f_c with a reasonable precision. Even a detailed analysis of the algorithms and codes can fail in many practical cases, when the code has multiple logical switches and completely different algorithms and computational schemes are used for different initial conditions and computational parameters. Estimation of the resource-specific parameter μ also poses a challenge on heterogeneous Grid resources, since there is a multitude of processors with the ratio of communication to computation performance spanning a few orders of magnitude. Moreover, the Grid exhibits dynamic network and processor performance; therefore, static domain decomposition is not applicable. To ensure efficient load balancing of a parallel application on the Grid, it is necessary to estimate β experimentally. There are two possible approaches to that: (1) directly measure the lumped value of β for the application on the allocated resources and (2) separately benchmark the resources, estimate μ and then find out the application-specific parameter f_c that would provide an optimal workload distribution on a given set of resources. The first approach requires serious intrusion into the application code. This is certainly not desirable, especially when targeting to build a generic load-balancing system which tries to abstract from the application specifics. We have chosen the second approach, for it is more generic and requires minimal modifications in the application code.

We have developed a meta-algorithm for AWLB on heterogeneous resources based on benchmarking the resource performance (defined as a set of individual resource parameters $\boldsymbol{\mu} = \{\mu_i\}$) and experimental estimation of the application parameter f_c. The target is to find the value of f_c minimizing the execution time, i.e. providing the best workload distribution for the application mapped to the resources characterized by the parameter set $\boldsymbol{\mu}$.

The outline of the load-balancing meta-algorithm is as follows:

1. Benchmark the resources dynamically assigned to the parallel application; measure the resource characteristics that constitute the set of resource parameters $\boldsymbol{\mu}$ (available processing power, memory and links bandwidth).
2. Estimate the range of possible values of the application parameter f_c. The minimal value is $f_c^{\min} = 0$, which corresponds to the case when no communications occur between the parallel processes of the application. The upper bound can be calculated based on the following reasoning: For the parallel processing to make sense, that is to ensure that running a parallel program on several processors is faster than sequential execution, the calculation time should exceed communication time. For homogeneous resources this can be expressed as follows:

$$\frac{T_{\text{comm}}}{T_{\text{calc}}} < 1 \quad \Leftrightarrow \quad \frac{N_{\text{comm}} t_{\text{comm}}}{N_{\text{calc}} t_{\text{calc}}} < 1 \quad \Leftrightarrow \quad f_c^{\max} = 1/\mu.$$

   Analogously, for heterogeneous resources the upper limit can be found as

$$f_c^{\max} = \max(t_{\text{calc}}^i)/\min(t_{\text{comm}}^i).$$

3. Search through the range of possible values of f_c in $[0 \ldots f_c^{\max}]$ to find the optimal value f_c^* minimizing the application execution time. For each value of f_c calculate the corresponding load distribution based on the resource parameters $\boldsymbol{\mu}$ determined in step 1 (details on calculating the load distribution weights will follow this algorithm). With this distribution perform one time step (iteration), and measure the execution time—the target optimization function. Selection of the next value of f_c can be done by any optimization method for unimodal smooth functions; for instance a simple line-search method can be used.
4. Execute further calculations using the discovered f_c^*.
5. In the case of dynamic resources where performance is influenced by other factors (which is generally the case on the Grid), a periodic re-estimation of resource parameters $\boldsymbol{\mu}$ and load re-distribution shall be performed during run-time of the application. Re-balancing shall be invoked if the application performance over the last step drops more than a certain user-defined threshold (expressed as a relative change in the execution time).
6. If the application is dynamically changing (for instance due to adaptive meshes, moving interfaces or different combinations of physical processes modelled at different simulation stages) then f_c^* must be periodically re-estimated on the same set of resources.

Periodic re-estimations in steps 5 and 6 can be easily organized for iterative, time-stepping or discrete-event simulations. After each step (iteration) the resource characteristics are automatically updated, and in case of significant application performance drop (below the user-defined threshold), the next step starts with an adapted load distribution. For other types of applications (continuous and not divided into logical steps), load re-balancing can be organized via check-pointing, which is a necessary capability for efficient fault-tolerant computing on the Grid.

The combination of $\boldsymbol{\mu}$ and f_c^* determines the distribution of the workload between the processors. To calculate the amount of workload per processor, we assign a weight factor to each processor according to its processing power, memory and network connection. A similar approach was applied in [23,25] for heterogeneous computer clusters, but the mechanism for

Please cite this article as: V.V. Korkhov, et al., A Grid-based Virtual Reactor: Parallel performance and adaptive load balancing, J. Parallel Distrib. Comput. (2007), doi: 10.1016/j.jpdc.2007.08.010

adaptive calculation of the weights and application requirements was not developed there. Moreover, the tools developed for cluster systems cannot be used in Grid environments without modifications since static resource benchmarking is not suitable for dynamic Grid resources, where the weights shall be calculated every time the solver is started on a new set of dynamically assigned processors.

Let us assume that for the ith processor: p_i is the available processor performance (e.g. in Flop/s), m_i is the available memory (in MB) and n_i is the available network bandwidth to the processor (in MB/s). An individual resource parameter μ_i then can be represented using the values of p_i, m_i, n_i. In a simple case when memory is considered only a constraining factor (and not driving the load-balancing process) it is $\mu_i = p_i/n_i$. This resource parameter is widely used in scientific applications where the most important factor is the ratio of the computational power to the network bandwidth. In a more general case, two parameters shall be considered, μ_i and m_i. And for the memory-driven applications, the ratio of the available memory to the network capacity of that processor m_i/n_i should play a major role in resource evaluation.

To reflect the processor capacity, we introduce a weighting factor w_i for each processor. It determines the final workload for a processor given by: $W_i = w_i W$, where W is the total workload.

To derive an expression for the weighting factors, we introduce parameters c_p, c_m and c_n that reflect computational, memory and communication requirements of the application. Then the weight of each processor is calculated using the following expression:

$$w_i = c_p p_i + c_m m_i + c_n n_i \text{ normalized to ensure that } \sum_i w_i = 1. \tag{2}$$

This weighting factor w_i reflects a relative capacity of the resources according to the measured infrastructure parameters $\mu_i = \mu(p_i, m_i, n_i)$ and the application parameter f_c. The infrastructure parameters μ_i can be initially estimated by a set of benchmarks before the actual calculations start (but after the resources have been assigned to the application). Searching through f_c with fixed values of μ_i gives us the optimal value f_c^* which corresponds to the optimal mapping of the workload to the resources.

Parameters c_p, c_m and c_n depend not only on the application characteristics but also on the heterogeneity of the resources. Let us analyse how these parameters and weighting factors w_i are related to f_c and μ_i. Consider a traditional situation when memory is only a constraining factor ($c_m = 0$). Then parameters c_p and c_n shall be proportional to the amount of application communications (computations) and the heterogeneity factors:

$$c_p = N_{\text{calc}} \varphi_{\text{proc}}, \quad c_n = N_{\text{comm}} \varphi_{\text{net}}. \tag{3}$$

Here φ_{proc} and φ_{net} are the heterogeneity metrics of processors and network links. In case of equal network links the weighting should be done only according to the processors capacity; therefore, the network heterogeneity parameter is nullified: $\varphi_{\text{net}} = 0$. Analogously, for homogeneous processors $\varphi_{\text{proc}} = 0$. The heterogeneity metrics of the network and computing resources can be expressed as a standard deviation of the set of normalized dimensionless resource parameters:

$$\varphi_{\text{net}} = \frac{\sum_{i=1}^{N} (n_i - n_{\text{avg}})^2}{N n_{\text{avg}}^2}, \quad \varphi_{\text{proc}} = \frac{\sum_{i=1}^{N} (p_i - p_{\text{avg}})^2}{N p_{\text{avg}}^2}.$$

Substituting c_p and c_n in Eq. (2) with expressions (3), the weights can be re-written as

$$w_i = N_{\text{calc}} \varphi_{\text{proc}} p_i + N_{\text{comm}} \varphi_{\text{net}} n_i.$$

In trivial cases,

$\varphi_{\text{net}} = 0$ (the network is homogeneous):

$$w_i = N_{\text{calc}} \varphi_{\text{proc}} p_i \sim p_i,$$

$\varphi_{\text{proc}} = 0$ (the processors are homogeneous):

$$w_i = N_{\text{comm}} \varphi_{\text{net}} n_i \sim n_i,$$

otherwise

$$\begin{aligned} w_i &= N_{\text{calc}} \varphi_{\text{proc}} p_i + N_{\text{comm}} \varphi_{\text{net}} n_i \\ &= N_{\text{calc}} \varphi_{\text{proc}} (p_i + n_i f_c \varphi_{\text{net}}/\varphi_{\text{proc}}). \end{aligned}$$

Defining $\varphi = \varphi_{\text{net}}/\varphi_{\text{proc}}$ as an aggregated heterogeneity metric of resources, keeping in mind that $\mu_i = p_i/n_i$ and omitting the constant multiplier $N_{\text{calc}} \varphi_{\text{proc}}$ before the brackets (which will be cancelled while calculating the normalized dimensionless weights), yields $w_i = p_i(1 + f_c \varphi/\mu_i)$. Introducing $\vartheta_i = \mu_i/\varphi$ that combines the characteristics of resource performance and heterogeneity, we get $w_i = p_i(1 + f_c/\vartheta_i)$, and the normalized dimensionless weights will be

$$w_i' = w_i \Big/ \sum w_i. \tag{4}$$

Knowing the fractional overhead of the application and the heterogeneity level of the resources, we can optimize the workload distribution using this fast weighting technique.

To evaluate the efficiency of the workload distribution we introduce the load-balancing speedup Θ as

$$\Theta = \frac{T_{\text{non–balanced}}}{T_{\text{balanced}}} \cdot 100\%, \tag{5}$$

where $T_{\text{non-balanced}}$ is the execution time without load balancing, and T_{balanced} is the execution time using load balancing on the same set of resources (the time taken to execute the algorithm itself is included). This metric is used to estimate the f_c^* that provides the best performance on given resources, i.e. the largest value of Θ in a given range of f_c. In a non-trivial case we expect to find a maximum of Θ and thus an optimal f_c^* for some workload distribution. Finite non-zero value of f_c^* means that the application requirements best fit the resources in this particular workload distribution, which minimizes the total runtime of the application. The case of $f_c^* = 0$ while $\varphi \neq 0$ means

Please cite this article as: V.V. Korkhov, et al., A Grid-based Virtual Reactor: Parallel performance and adaptive load balancing, J. Parallel Distrib. Comput. (2007), doi: 10.1016/j.jpdc.2007.08.010

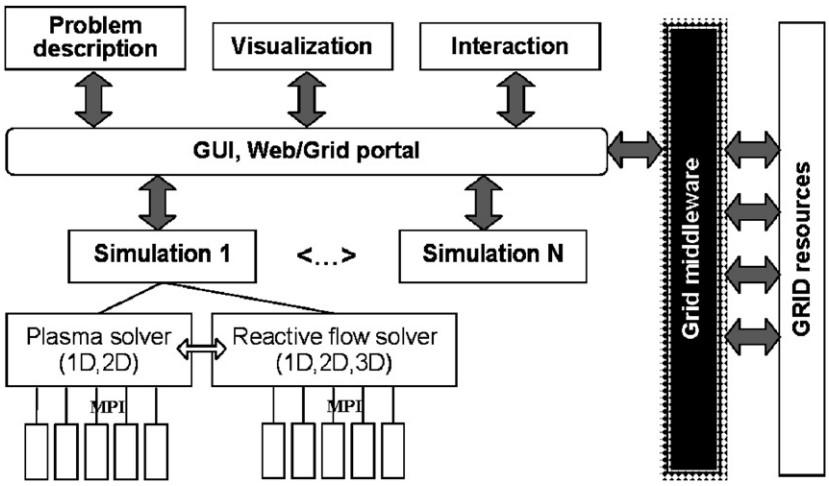

Fig. 1. Functional scheme of the Virtual Reactor application.

that the application is computation dominated, i.e. the amount of communications is negligible, and the optimal workload distribution will be proportional only to the computational power of the processors. $\varphi_{\text{net}} = 0$ means that we consider heterogeneous processors connected by homogeneous network links. In this case the value of f_c does not influence the distribution, which shall be proportional only to the processing power.

In the discussion presented above while deriving Eq. (4), we considered a simple case when memory requirements put only a Boolean constraint to the allocation of processes on the resources: either there is enough memory to run the application or not. But it can play a role in the load-balancing process, being one of the determining factors of application performance. This is the case for applications that are able to control memory requirements according to the available resources. In this case there will be additional parameters analogous to f_c and μ_i (or these functions will be more complex), but the idea and the load-balancing mechanism remain the same.

3. Case study on adaptive load balancing: the Virtual Reactor

3.1. The Virtual Reactor overview and its implementation on the Grid

A complex PSE usually has a modular architecture and consists of a number of loosely or tightly coupled components [28]. Our test case, the Virtual Reactor, includes the basic components for reactor geometry design; computational mesh generation; plasma, flow and chemistry simulation; editors of chemical processes and gas properties connected to the corresponding databases; pre- and postprocessors, visualization and archiving modules [15]. The aim of our research is to virtualize separate modules of the application to run them efficiently as services and access them on the Grid.

The application components perform one (or a few) of the following functions: problem description, simulation, visualization and interaction. This is schematically shown in Fig. 1, where we emphasize the *simulation* components.

The core components are modules simulating plasma discharge, gas flow, chemical reactions and film deposition processes occurring in a PECVD reactor. The details on numerical methods and parallel algorithms employed in the solvers are described in [12]. The most important features relevant to the Grid implementation are as follows: for stability reasons, implicit finite volume schemes were applied, thus forcing us to use a sweep-type algorithm for solving equations in every "beam" of computational cells in each spatial direction of the Cartesian mesh. A special parallel algorithm was developed with beams distributed among the processors. Communications are organized exploiting a Master–Slave model, where at each simulated time step the Master prepares instructions for the Slaves, sends them the data to be processed, receives the results and processes them before proceeding to the next step. The algorithm was implemented in an SPMD model, using the MPI message passing interface with MPI Barrier points for synchronization. Data exchange between the Master and the Slaves is repeated every time step, and simulation proceeds for thousands to millions time steps. In the testbed we use generic MPICH-P4 built binaries that can be executed on all the testbed machines using the Globus job submission service. To study the influence of various parameters on the simulated processes we run a number of simulations in parallel (shown in Fig. 1 as "Simulation 1" … "Simulation *N*" blocks) with the assistance of Nimrod-G [20].

To provide efficient execution of a parallel application on heterogeneous resources, it is necessary to first test the application performance dependencies on homogeneous resources. This gives an insight into the application scalability, induced fractional overhead, dependency of the communication and calculation time on the number of processors used, etc. The results of such tests can help estimating and predicting the behaviour of the application on heterogeneous resources, thus simplifying the adaptation process.

Please cite this article as: V.V. Korkhov, et al., A Grid-based Virtual Reactor: Parallel performance and adaptive load balancing, J. Parallel Distrib. Comput. (2007), doi: 10.1016/j.jpdc.2007.08.010

3.2. RDG testbed infrastructure

Generally the infrastructure of a site within a Grid testbed can be of one of the following types depending on the underlying resources:

(I) traditional homogeneous computer cluster architecture: homogeneous worker nodes and uniform interconnection links;
(II) homogeneous worker nodes with heterogeneous interconnections;
(III) heterogeneous worker nodes with uniform interconnections;
(IV) heterogeneous nodes with heterogeneous interconnections.

A complete Grid infrastructure is always of the Type IV, characterized by severe heterogeneity with a wide range of processor and network communication parameters. As we show later in this paper, the type of resources allocated to a parallel application significantly influences its performance, and different load-balancing techniques shall be applied to different combinations of the resources.

Currently the RDG testbed consists of six sites with different infrastructures: Amsterdam-1 (contains 3 nodes, 4 processors)—Type IV; Amsterdam-2 (32 nodes, 64 processors)—Type I; St. Petersburg (4 nodes, 6 processors)—Type IV; Novosibirsk (4 processors)—Type II; Moscow-1 (13 nodes, 26 processors)—Type I; Moscow-2 (12 nodes, 24 processors)—Type I.

The RDG testbed is built with the CrossGrid middleware [26] based on the LCG-2 distributions and sustains the interoperability with the CrossGrid testbed. More detailed information on the RDG testbed can be found in [22]. The RDG Virtual Organization (VO) is included into the CrossGrid VO, thus allowing the RDG certificate holders to access the CrossGrid resources and services. The CrossGrid testbed consists of 16 sites with the infrastructures of all four types and offers over 400 processors.

4. Application performance on homogeneous sites

4.1. Testing approach

Benchmarking a complex application is required to evaluate its performance and reveal the dependencies of its behaviour on the underlying infrastructure. We use a structural approach to benchmark the Virtual Reactor. Within this approach, the overall functionality of the whole system is studied, followed by performance measurements of the individual components while they are not influenced by activities of the other components.

Benchmarking the components allows evaluating their performance depending on various parameters like input data and the resources used. This helps to predict the performance of a given component and organize efficient resource allocation, thus improving the overall resource management within the whole application.

The earlier tests of the Virtual Reactor performed on the CrossGrid testbed showed that most of the interactive components of the Virtual Reactor do not put restrictions on the computer systems and network bandwidth and can be efficiently executed on distributed Grid resources [15]. Next, we focused on benchmarking the *simulation* modules. Each simulation consists of two basic components: one for plasma simulation and another for reactive flow simulation (see Fig. 1). These two components exchange only a small amount of data every hundred or thousand time steps; therefore, the network bandwidth is not critical for their communication. Finally, we concentrate on benchmarking the individual parallel solvers, starting from a 2D PECVD solver which maintains all the features of the 3D one, but takes less time to estimate the solver behaviour on the Grid.

4.2. Test setup

The goal of our benchmarks is to determine the scalability of the application, find out the limitations on the efficiency posed by the application architecture, resources and types of the simulations. Uncovering such details will allow us to optimize resource management strategy for allocating the application components within the whole Virtual Reactor PSE.

The solver operates a reactor geometry that is composed of a number of connected blocks. Different types of simulation can be performed within a single geometry: a chemically inactive flow and a flow with chemical and plasma processes. Physically the problem type is determined by the gas mixture composition, temperatures, pressures and the plasma discharge operation mode. From the computational point of view these types of simulations differ by the ratio of computations to communications: in case of simulating chemical processes the computational load is significantly higher.

We started from a light-weight problem not simulating the chemical and plasma processes, with a simplified reactor geometry consisting of a single block that allows easy tracking of parameter influence on the execution time. To measure the dependency of the solver performance upon the input data, multiparameter variation has been applied. We measured the solver execution time, speedup and communication time depending on the combinations of input parameters: the computational mesh size, number of simulation time steps and number of processors.

The benchmark tests had to be automated because the parameter variation leads to a large number of job submissions. To solve this problem we have built an execution environment to support series of parameter-sweep Globus job submissions. The environment is generic and can be used for any kind of performance benchmarks with user-defined metrics and parameters to be analysed. Within this environment, the application to benchmark is described using some templates that are filled with particular application data (e.g. Globus RSL template for job submission which also contains the list of input and output files). One of the functionalities of this execution environment is the support for parameter-sweep runs, analogous to what

Please cite this article as: V.V. Korkhov, et al., A Grid-based Virtual Reactor: Parallel performance and adaptive load balancing, J. Parallel Distrib. Comput. (2007), doi: 10.1016/j.jpdc.2007.08.010

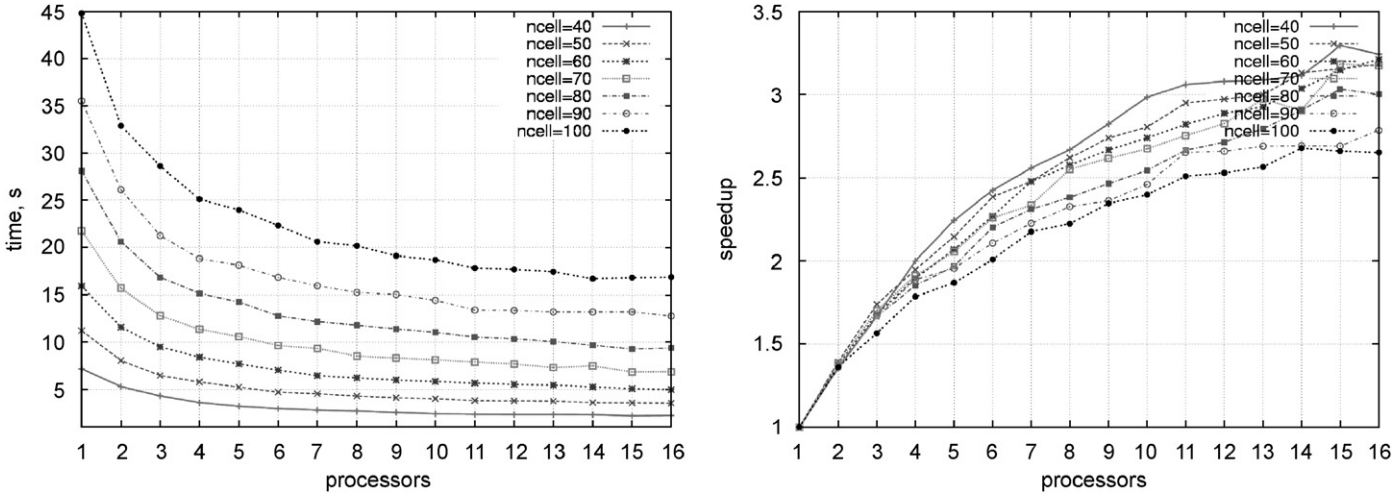

Fig. 2. Light-weight (no chemistry) simulation: total execution time and speedup for different computational mesh sizes.

Nimrod-G or Condor-G provides. The advantage of our implementation is that we can specify the parameters (and their ranges) that shall be changed, as well as the characteristics to be measured and visualized automatically to analyse the influence of those parameters.

In these tests, a single-block topology was used. The block was subdivided into a (*ncell* × *ncell*) number of computational mesh cells, with *ncell* running from 40 to 100, thus forming 1600–10 000 cells. We also performed some tests with real reactor geometries in order to check whether reactor topology influences parallel performance, since potentially it can introduce some load imbalance.

4.3. Influence of the number of time steps and reactor topology

Experiments with a different number of time steps showed that the execution time and other measured parameters are linearly proportional to the number of time steps, provided that this number is high enough and the standard output and hard disk operations are kept minimal (that means no excessive logging, no storing the 2D fields or other additional files every time step). All the results presented below are measured for 100 time steps.

Along with the single-block geometry, we studied the performance of the solver with a complex multi-block PECVD reactor topology, which consists of an equivalent number of computational mesh cells. The results showed that all the measured characteristics (execution time, speedup, computation and communication time) on the same resources coincide for the single-block and multi-block topologies of equal number of cells within 1% accuracy. This assures us that the parallel algorithm used in the solver provides a good load balancing even in cases of complex topologies. Further we test the influence of the problem size (the number of mesh cells) with the single-block reactor geometry, since it is easier to vary the mesh size arbitrarily with a single-block geometry than with a multi-block complex topology.

4.4. Speedup of the chemistry-disabled and chemistry-enabled simulations

The measurements were carried out on all the Grid sites within the RDG testbed. The parallel solver showed a noticeable speedup on the Moscow and Amsterdam sites of Type I (homogeneous cluster with uniform communication links). Figs. 2 and 3 demonstrate the total execution time and speedup of the parallel solver for different types of simulation: A chemistry-disabled "light-weight" simulation (Fig. 2) and a chemistry-enabled "heavy" simulation (Fig. 3).

We observe different trends of the solver performance: for the light-weight simulation the speedup decreases with the increase in the mesh size (see the different curves in Fig. 2, right), while for the chemistry-enabled simulation the speedup increases with the problem size (Fig. 3). Different trends in the speedup dependency on the problem size are discussed and explained in detail in Sections 4.6 and 4.7.

The same parallel solver tested on homogeneous Grid sites with a higher ratio of the inter-process communication bandwidth to the processor performance achieved much higher speedups, for instance on lisa.sara.nl with Infiniband interconnections it was 3 times higher for the large problem size simulations. The type of MPI library also influences the parallel efficiency of a program: a specialized library optimized for the native communication technology (e.g. MPICH-GM for Myrinet communications on das2.nikhef.nl) increases the speedup up to two times compared to the generic MPICH-P4 or MPICH-G2.

4.5. Communication time trends

The time spent on inter-process communications within the solver is shown in Fig. 4 for different mesh sizes. The communication time was calculated as a sum of MPI Send/MPI Receive time on the master node over the total number of iterations.

Please cite this article as: V.V. Korkhov, et al., A Grid-based Virtual Reactor: Parallel performance and adaptive load balancing, J. Parallel Distrib. Comput. (2007), doi: 10.1016/j.jpdc.2007.08.010

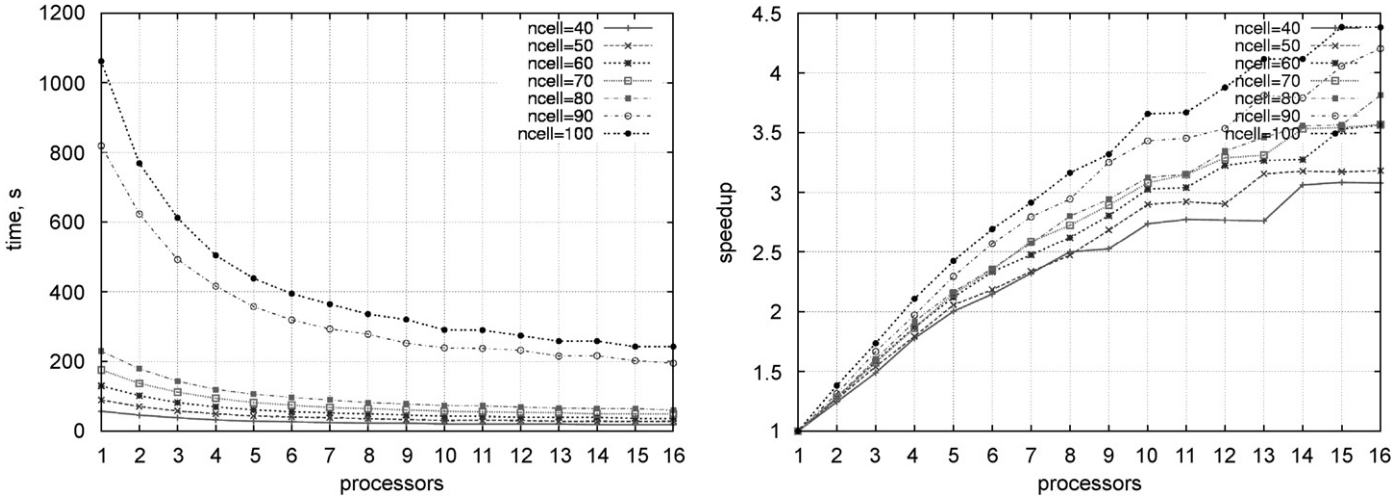

Fig. 3. Chemistry-enabled simulation: total execution time and speedup for different computational mesh sizes.

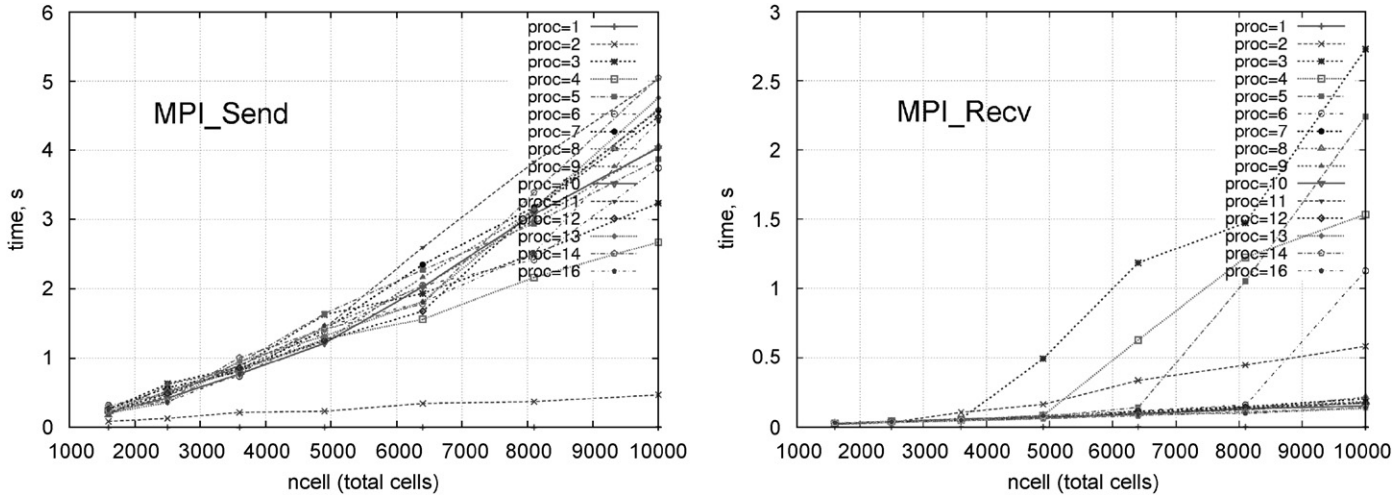

Fig. 4. Dependency of the communication time on the computational mesh size for different numbers of processors "*proc*". Light-weight simulation.

We observe that communication time grows super-linearly with the mesh size increase, although the amount of data is linearly proportional to the number of mesh cells. This will be discussed in Section 4.7. The fact that the MPI_Recv time is less than the MPI_Send time is explained by the amount of data sent from the slaves to the master: it is approximately half of what the master sends to the worker nodes.

If we plot the communication time versus the number of processors, then some peculiarities can be observed (see Fig. 5): (1) The communication time grows non-monotonically with the number of processors, but drops down a little with an even number of processors and (2) the time of MPI Receive calls is an order of magnitude higher for the larger meshes on the first few processors. These observations are discussed in Section 4.7.

4.6. Computation to communication ratio

In Fig. 6 the total execution time is presented along with the contributions of calculation and communication. For a smaller computational mesh (Fig. 6, left), the communication time makes a relatively small contribution to the total execution time even for a large number of processors involved. For a larger mesh (Fig. 6, right), communication makes up to 30% of the execution time. This result confirms that the network bandwidth is not sufficient for this type of problem (see also the explanations to Fig. 3).

As it was mentioned in the previous Section, the solver can simulate the chemical and plasma processes within the reactor along with the gas flow. Fig. 7 demonstrates the ratio of computation to communication time for different mesh sizes with different types of the simulation. The higher the ratio is the less communications are required, which obviously offers a better parallel efficiency and application scalability. The ratios in Fig. 7 explain the different speedup trends observed in Figs. 2 and 3 for chemistry-enabled and chemistry-disabled (light-weight) simulations. From the presented graphs we can see that the behaviour of this ratio does not depend on the mesh size for the chemistry-enabled simulations, while this behaviour

Please cite this article as: V.V. Korkhov, et al., A Grid-based Virtual Reactor: Parallel performance and adaptive load balancing, J. Parallel Distrib. Comput. (2007), doi: 10.1016/j.jpdc.2007.08.010

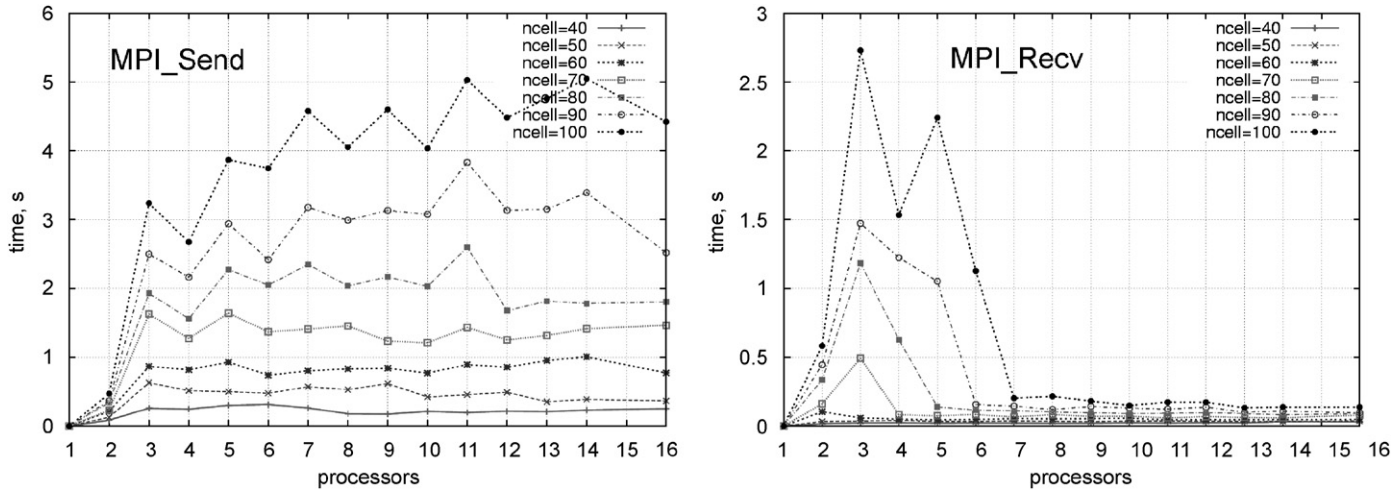

Fig. 5. Dependency of the communication time on the number of processors for different computational mesh sizes. Light-weight simulation.

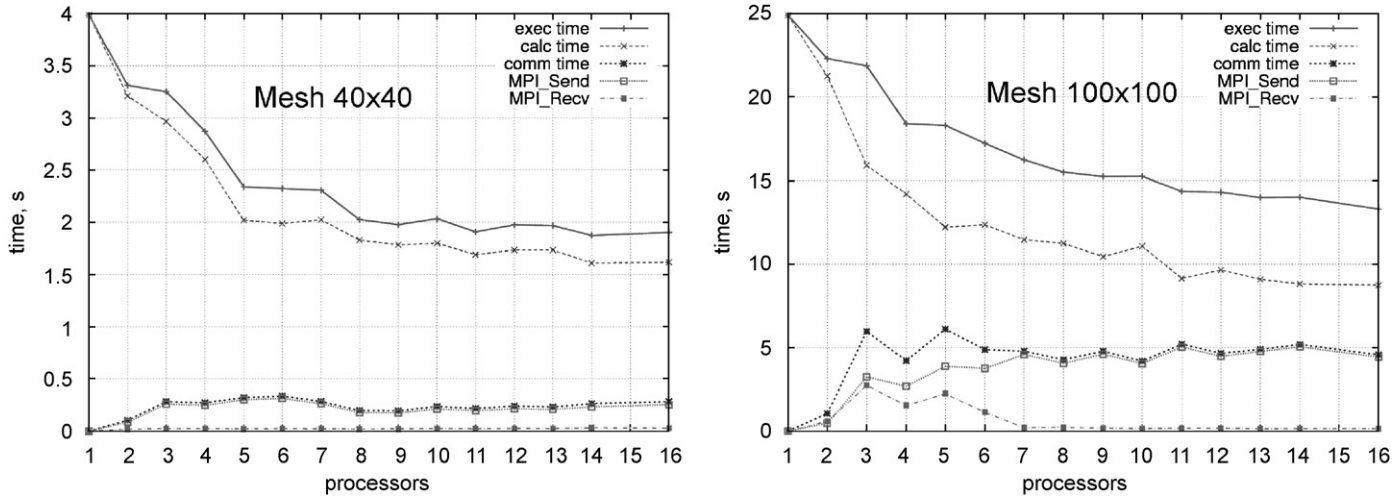

Fig. 6. Total execution time and contributions of the calculation and communication depending on the number of processors for different computational mesh sizes (light-weight simulation).

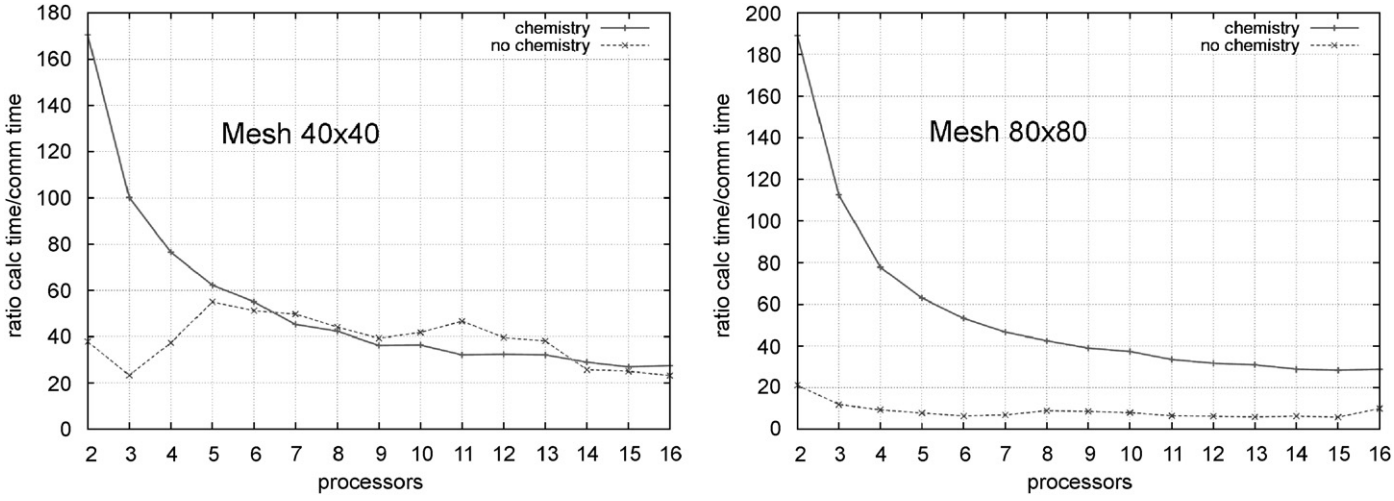

Fig. 7. The ratio of the computation to communication time for chemistry-enabled and light-weight simulations.

Please cite this article as: V.V. Korkhov, et al., A Grid-based Virtual Reactor: Parallel performance and adaptive load balancing, J. Parallel Distrib. Comput. (2007), doi: 10.1016/j.jpdc.2007.08.010

for the light-weight simulations significantly differs for small and large mesh sizes. For a small mesh size, the ratio stays high, and for 6 processors and more it reaches the level of the chemistry-enabled simulations. For a larger mesh, the computation/communication ratio for the simulations with no chemistry is very low, thus diminishing the overall parallel efficiency.

4.7. Discussion of the results for homogeneous resources

The results presented in Section 4.4 show that the parallel speedup is lower for a larger problem size for the simulations with no chemistry (see Fig. 2). This fact indicates that the ratio of the inter-process communication bandwidth to the processor performance was not high enough for light-weight problems with relatively small number of operations per computational cell. It means that for an optimal usage of computing power, a large number of processors for one parallel run shall only be used for relatively small computational meshes. Thus the communication technology puts a limit to the scalability of the solver for this problem type. On the other hand, the simulation of the flow *with* chemical processes shows higher speedup with larger meshes (see Fig. 3). Here the amount of computations brought by simulating the chemistry changes the behaviour of the solver qualitatively. This leads us to the conclusion that different resource allocation strategies should be applied for different types of simulation and meshes used.

The results in Fig. 5 reflect the network and node features of the tested Grid site:

1. Since the site consists of dual nodes, the network channels work more efficiently for data transfers between the Master and a Slave processor if a connection was already established with another Slave processor on the same node. This can be explained by implementation of the MPI library which saves network resources while opening and maintaining connections for concurrent processes on the same node.
2. The "peaks" of the MPI Receive time for the first few processors (see Fig. 5, right) are caused by the constraints on the portions of data that could be accommodated at once. The constraining factors could be the network bandwidth distribution, the processor cache size, the memory available on the node or a combination of these factors.

5. Application performance on heterogeneous resources

5.1. Performance of the original parallel solver on heterogeneous resources

The RDG sites with heterogeneous processors and/or network links (Types II, III, IV) provided only a limited parallel speedup or even a slow-down of the original solver with a homogeneous parallel algorithm (data not shown). This was inevitable since in addition to the low-bandwidth links, these sites are characterized by very diverse resources: the processor and network parameters differ by orders of magnitude.

The parallel algorithm used in the solver was originally developed for homogeneous computer clusters with equal processing power, memory and inter-processor communication bandwidth. In case of submitting equal portions of a parallel job to the nodes with different performance, all the fast processors have to wait at the barrier synchronization point till the slowest ones catch up, thus the effect of slow-down on heterogeneous resources is not surprising. The same problem occurs if the network connection from the Master processor to some of the Slave processors is much slower than to the others. As we have shown in the previous section, for communication-bound simulations (chemistry-disabled simulation with large computational meshes), the communication time on low-bandwidth networks is of the order of the calculation time; therefore, the heterogeneity of the inter-processor communication links is a hindrance as considerable as the diversity of the processing power. One of the natural ways to adapt the solver to the heterogeneous Grid resources is to distribute the portions of job among the processors according to the processor performance and network connections, taking into account the application characteristics. To adapt the parallel solver, we applied the approach presented in Section 2.

5.2. Experimental results of the workload-balancing algorithm

To illustrate the approach described in Section 2 we present the results obtained for different types of simulation (chemistry-disabled and enabled) of a reactor geometry with 10 678 cells on the St. Petersburg Grid site. This site is heterogeneous in both the CPU power and the network connections of the processors (Type IV). There are two 1.8 GHz nodes (nwo1, nwo2) and two dual 450 MHz nodes (crow2, crow3), all having 512 MB RAM. One of the dual nodes (crow3) is placed in a separate network segment with 10 times lower bandwidth (10 Mbit/s against 100 Mbit/s in the main segment). The load-balancing tests were performed with a moderate-size problem which does not pose restrictions on required memory, thus the memory influence parameter c_m was reduced to zero and the exploration was done for the application parameter f_c. The link bandwidth between the Master and Slave processors was estimated by measuring the time of MPI_Send transfers of a predefined data block (with the MPI buffer size equal to 10^6 of MPI_DOUBLEs) during the solver execution, after the resources have been allocated. In these measurements the same logical network topology was used as employed in the solver. The CPU power and available memory were obtained by a function from the *perfsuite* library [17]. To validate the approach presented in Section 2 we applied the workload-balancing technique for a single simulation running on different sets of heterogeneous resources. The estimation of performance for different possible values of the parameter f_c (hence, different weighting and workload distribution) was carried out. For one simulation type we expect to obtain approximately the same value of the parameter f_c^* (that provides the best performance, see Section 2) on different sets of resources. Fig. 8 (left) illustrates the load-balancing speedup Θ achieved by applying the

Please cite this article as: V.V. Korkhov, et al., A Grid-based Virtual Reactor: Parallel performance and adaptive load balancing, J. Parallel Distrib. Comput. (2007), doi: 10.1016/j.jpdc.2007.08.010

Table 1
Distribution of processors and balancing weights providing the best load-balancing speedup for different sets of resources.

Sets of resources	Weights assigned to each processor						Heterogeneity metrics		Best balancing speedup
	nwo1, 1.8 GHz/ 100 Mb/s	crow2/1, 450 MHz/ 100 Mb/s	crow3/1, 450 MHz/ 10 Mb/s	crow2/2, 450 MHz/ 100 Mb/s	nwo2, 1.8 GHz/ 100 Mb/s	crow3/2, 450 MHz/ 10 Mb/s	φ_{proc}	φ_{net}	Θ (%)
Set I, 3 processors	0.580	0.274	0.146	–	–	–	0.618	0.606	196
Set II, 4 processors	0.452	0.218	0.112	0.218	–	–	0.638	0.502	182
Set III, 5 processors	0.314	0.146	0.080	0.146	0.314	–	0.591	0.439	201
Set IV, 6 processors	0.278	0.160	0.062	0.160	0.278	0.062	0.618	0.606	207

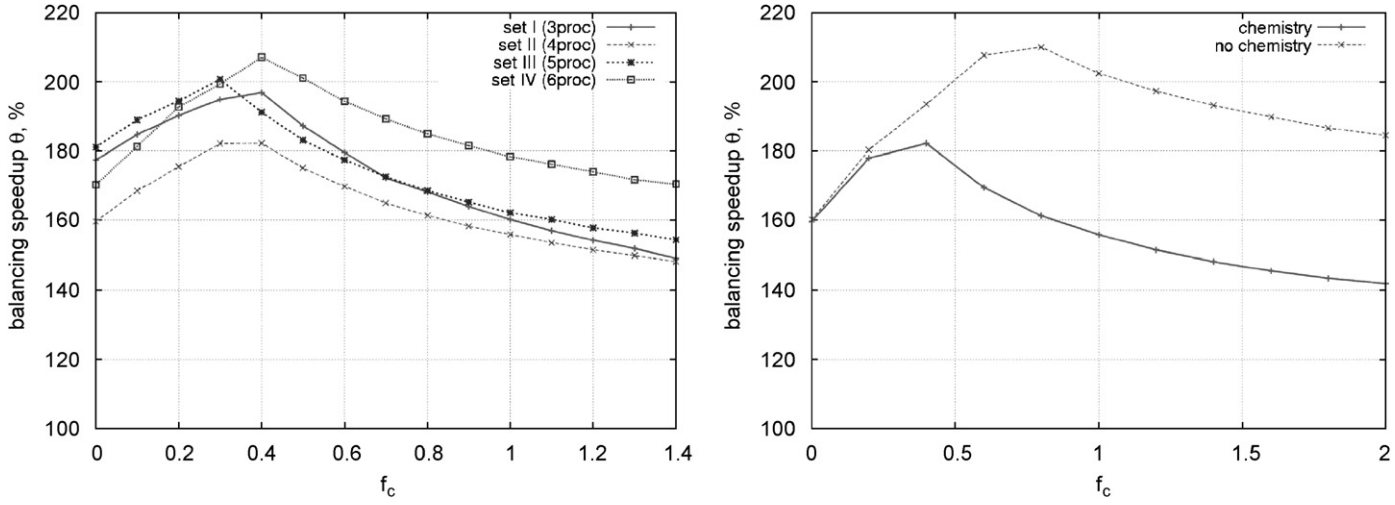

Fig. 8. Dependency of the balancing speedup Θ on the parameter f_c. Left: single simulation on different sets of resources. Right: different types of simulation on one set of resources.

workload-balancing technique for different values of the parameter f_c on several fixed sets of heterogeneous resources for a light-weight (chemistry-disabled) simulation. In Table 1 we summarize the combinations of processors dynamically allocated in four tests (different sets of resources) and the weights assigned to each processor for the values of f_c^* providing the best execution time, thus the maximal balancing speedup (see Fig. 8, left).

Fig. 8 (left) shows that for a given simulation the best performance is delivered by weighting the resources with the value of $f_c \approx 0.3$–0.4. Noticeably, this corresponds to the value obtained for this simulation during the preliminary analysis on homogeneous resources (compare to results for similar simulations in Section 4.6, Fig. 7). The algorithm increases the speedup up to 207% compared to the initial non-balanced version of the code on the tested sets of resources. We can see that the distribution of the workload proportional only to the processing power ($f_c = 0$) also gives a significant increase in the performance, but introduction of the dependency on application-specific communication/computation ratio f_c and resource infrastructure parameters μ_i adds another 40% to the balancing speedup Θ.

Fig. 8 (right) shows the dependency of the balancing speedup Θ for different types of simulation (chemistry-enabled and chemistry-disabled) on the same set of resources (set III from Table 1). The chemistry-disabled simulation has a higher communication/computation ratio (as was shown also in Section 4.6, Fig. 7). This is clearly seen in the experimental results where chemistry-disabled simulation obtains the highest balancing speedup Θ at higher f_c values. Moreover, the gain in the balancing speedup (maximal value of Θ) is higher for the simulation with a larger fraction of communications. These results illustrate that the introduced algorithm for resource AWLB can bring a valuable increase in the performance for communication-intensive parallel programs running on heterogeneous resources.

Please cite this article as: V.V. Korkhov, et al., A Grid-based Virtual Reactor: Parallel performance and adaptive load balancing, J. Parallel Distrib. Comput. (2007), doi: 10.1016/j.jpdc.2007.08.010

5.3. Discussion and suggestions for generalized automated load balancing

The introduction of the load-balancing technique allowed us to increase the efficiency of the parallel solver on heterogeneous resources. The proposed method of successive benchmarking the resource infrastructure parameters μ_i and estimating the application parameter f_c shows the possibility of automatic load balancing for applications whose internal structure (computations and communications) is not known.

Analysis of the results achieved with the workload-balancing algorithm suggested that the following issues shall be addressed in order to optimize the balancing technique:

1. To measure the inter-process communication rate, we sent a fixed amount of data from the Master to each Slave processor. However, in some cases the response of the communication link's to the increasing amount of data is not linearly proportional as shown in Fig. 4. For the slower networks this tendency is even more pronounced. This brings us to a conclusion that the amount of data sent to measure the link's performance shall be close to the amount really transferred within the solver for every particular mesh size, geometry and solver type. Another option to estimate the inter-processor communication rate is to analyse the iteration data transfer time during the actual execution. However, this requires significant code modifications and might be undesirable.
2. To properly take into account the memory requirements of each particular instance of a parallel solver, similar reasoning shall be applied to the choice of the c_m coefficient as to the selection of c_p and c_n.
3. The specialty of the memory factor is that in addition to this resource dependency it is strongly influenced by the application features. To take into account the memory requirements of a parallel solver, the weighting algorithm must be enriched by the function measuring the memory requirements per processor for each simulation on each set of resources. In case of sufficient memory on allocated processors, the load balancing can be performed taking into account all the factors (CPU, memory and network) where memory factor is a constraint. After this, another check of meeting the memory requirements on each processor must be performed. In the unfavourable case of insufficient memory on some of the processors, they must be disregarded from the parallel computation or replaced by other, better-suited processors. This must be done preferably outside the application, on the level of job scheduling and resource allocation. This brings us to the conclusion that ideally a combined technique shall be developed, where the application-centred load-balancing approach is coupled with a system-level resource management.

6. Conclusions

One of the most challenging problems in porting parallel distributed applications from homogeneous cluster environments to heterogeneous resources is to keep up a high level of parallel efficiency of the computational components. To tackle this problem, we developed a theoretical approach and a generic workload-balancing technique that takes into account specific parameters of the resources dynamically assigned to a parallel job, as well as the application requirements. We validated the proposed algorithm by applying it to the Virtual Reactor parallel solvers running on the RDG testbed. It is worth noting that the load-balancing speedup goes through a maximum at $f_c = f_c^*$ as shown in Fig. 8. This indicates that the load-balancing strategy does find an optimum in the complex parameter space of the heterogeneous application/architecture combination. The clear maximum gives an unbiased guide towards automatic load balancing. The developed approach is well suited for either static or dynamic load balancing, and can be combined with the Grid performance prediction models or application-level scheduling systems [4,24].

In order to optimize the resource management strategy for the Virtual Reactor, we benchmarked the individual components on a set of diverse RDG resources, and extensively studied the behaviour of the parallel solvers with various problem types and input data on different resource infrastructures. The results clearly show that even within one solver different trends can exist in the application requirements and parallel efficiency depending on the problem type and computational parameters; therefore, distinct resource management and optimization strategies shall be applied, and automated procedures for load balancing are needed to successfully solve complex simulation problems on the Grid.

To further test the load-balancing algorithm, we have developed a synthetic application with tuneable characteristics. It allows one to model applications with different computation and communication requirements and logical network topologies. Some results of that work have been published in [13]. In [10,11] we compared the theoretically derived optimization parameters for some specific topologies of parallel applications with those predicted by our heuristic algorithm. Currently we are integrating the adaptive load-balancing algorithm with the DIANE user-level scheduling system, which will extend our testing ground to the multitude of real applications executed on the EGEE grid.

Acknowledgments

The authors would like to thank Irina Shoshmina, Alfredo Tirado-Ramos and the RDG Grid deployment team for their assistance. The research was conducted with financial support from the Dutch National Science Foundation NWO and the Russian Foundation for Basic Research under project numbers 047.016.007 and 047.016.018, and with partial support from the Virtual Laboratory for e-Science Bsik project [27].

References

[1] Available from: ⟨proj-openlab-datagrid-public.web.cern.ch⟩, ⟨www.nbirn.net⟩, ⟨www.fusiongrid.org⟩, ⟨www.globus.org/alliance/projects.php⟩, ⟨ca.sandia.gov, www.us-vo.org⟩.

[2] Available from: ⟨www.cfdrc.com⟩, ⟨www.fluent.com⟩, ⟨www.semitech.us⟩, ⟨www.softimpact.ru⟩.
[3] A. Barak, G. Shai, R. Wheeler, The MOSIX distributed operating system, load balancing for UNIX, Lecture Notes in Computer Science, vol. 672, Springer, Berlin, 1993.
[4] F. Berman, et al., Adaptive computing on the grid using appLeS, IEEE Trans. Parallel Distributed Systems 14 (4) (2003) 369–382.
[5] R. David, et al., Source code transformations strategies to load-balance grid applications, Lecture Notes in Computer Science, vol. 2536, Springer, Berlin, 2002, pp. 82–87.
[6] J.F. de Ronde, A. Schoneveld, P.M.A. Sloot, Load balancing by redundant decomposition and mapping, Future Generation Comput. Systems 12 (5) (1997) 391–407.
[7] J.F. de Ronde, Mapping in high performance computing. A case study on finite element simulation, Ph.D. Thesis, University of Amsterdam, 1998.
[8] G. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, D. Walker, Solving Problems on Concurrent Processors, vol. 1, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[9] K.A. Iskra, F. van der Linden, Z.W. Hendrikse, B.J. Overeinder, G.D. van Albada, P.M.A. Sloot, the implementation of dynamite—an environment for migrating PVM tasks, operating systems review, Association for Computing Machinery, Special Interest Group on Operating Systems, vol. 34(3), July 2000, pp. 40–55.
[10] V.V. Korkhov, V.V. Krzhizhanovskaya, Workload balancing in heterogeneous Grid environment: a virtual reactor case study. Proceedings of the Second International Conference on Distributed Computing and Grid Technologies in Science and Education. JINR, Dubna, 111-2006-167, ISBN 5-9530-0138-X, 2006, pp. 103–113.
[11] V. Korkhov, V. Krzhizhanovskaya, Benchmarking and adaptive load balancing of the Virtual Reactor application on the Russian–Dutch Grid, Lecture Notes in Computer Science, vol. 3991, Springer, Berlin/Heidelberg, 2006, pp. 530–538.
[12] V.V. Krzhizhanovskaya, et al., Distributed simulation of silicon-based film growth, Lecture Notes in Computer Science, vol. 2328, Springer, Berlin, 2002, pp. 879–888.
[13] V.V. Krzhizhanovskaya, V.V. Korkhov, Dynamic load balancing of black-box applications with a resource selection mechanism on heterogeneous resources of the Grid, Lecture Notes in Computer Science, vol. 4671, 2007, pp. 245–260.
[14] V.V. Krzhizhanovskaya, V.V. Korkhov, A. Tirado-Ramos, D.J. Groen, I.V. Shoshmina, I.A. Valuev, I.V. Morozov, N.V. Malyshkin, Y.E. Gorbachev, P.M.A. Sloot, Computational engineering on the Grid: crafting a distributed Virtual Reactor, Second IEEE International Conference on e-Science and Grid Computing (e-Science'06), Amsterdam, The Netherlands, December 4–6 2006, IEEE Computer Society Press, Silver Spring, MD, 2006, p. 101.
[15] V.V. Krzhizhanovskaya, P.M.A. Sloot, Yu.E. Gorbachev, Grid-based simulation of industrial thin-film production, Simulation: Trans. Soc. Modeling Simulation Internat. 81 (1) (2005) 77–85.
[16] V.V. Krzhizhanovskaya, M.A. Zatevakhin, A.A. Ignatiev, Y.E. Gorbachev, W.J. Goedheer, P.M.A. Sloot, A 3D virtual reactor for simulation of silicon-based film production, Proceedings of the ASME/JSME-PVP Conference, ASME PVP, vol. 491-2, PVP2004-3120, 2004, pp. 59–68.
[17] R. Kufrin, PerfSuite: An Accessible, open source performance analysis environment for Linux, 6th International Conference on Linux Clusters, Chapel Hill, NC, 2005.
[18] Z. Lan, V.E. Taylor, G. Bryan. Dynamic load balancing of SAMR applications on distributed systems, Proceedings of the ACM/IEEE Conference on Supercomputing, 2001.
[19] C. Lu, S.-M. Lau, An adaptive load balancing algorithm for heterogeneous distributed systems with multiple task classes, International Conference on Distributed Computing Systems (ICDCS'96), 1996.
[20] Nimrod-G: ⟨http://www.csse.monash.edu.au/∼davida/nimrod/⟩.
[21] G. Shao, R. Wolski, F. Berman, Master/slave computing on the grid, Proceedings of Heterogeneous Computing Workshop, IEEE Computer Society Press, Silver Spring, MD 2000, pp. 3–16.
[22] I. Shoshmina, et al., Experience of exploiting the RiDGrid segment. Proceedings of the Second International Conference on Distributed Computing and Grid Technologies in Science and Education, JINR, Dubna, D11-2006-167, ISBN 5-9530-0138-X, 2006.
[23] S. Sinha, M. Parashar, Adaptive runtime partitioning of AMR applications on heterogeneous clusters, Proceedings of 3rd IEEE International Conference on Cluster Computing, 2001, pp. 435–442.
[24] X.-H. Sun, M. Wu, Grid harvest service: a system for long-term, application-level task scheduling, Proceedings of IEEE International Parallel and Distributed Processing Symposium, 2003.
[25] J.D. Teresco, et al., Resource-aware scientific computation on a heterogeneous cluster, Comput. Sci. Eng., vol. 7(2), 2005, pp. 40–50.
[26] The CrossGrid EU Science project: ⟨http://www.eu-CrossGrid.org⟩.
[27] The Virtual Laboratory for e-Science project: ⟨http://www.vl-e.nl⟩.
[28] D.W. Walker, M. Li, O. Rana, M.S. Shields, Y. Huang, The software architecture of a distributed problem-solving environment, Concurrency—Practice Experience 12 (15) (2000) 1455–1480.
[29] Y. Zhang, K. Hakozaki, H. Kameda, K. Shimizu, A performance comparison of adaptive and static load balancing in heterogeneous distributed systems, Proceedings of the 28th Annual Simulation Symposium, 1995, p. 332.

Vladimir Korkhov is a PhD candidate at the Faculty of Science of University of Amsterdam. He received his Master's degree in mathematics and computer science from St. Petersburg Institute of Fine Mechanics and Optics, Russia. His research interests include grid computing, distributed software systems, resource management and workload balancing in heterogeneous environment, workflows on the grid; he is the author of more than 20 conference and journal papers.

Valeria Krzhizhanovskaya is a researcher at the University of Amsterdam (UvA), The Netherlands, and a senior lecturer at St. Petersburg State Polytechnic University (StPSPU), Russia. She received the MSc degree in Applied Mathematics and Physics from StPSPU and is finalizing her PhD in Computational Science at the UvA. Valeria has published over 40 papers, worked as a guest editor of 4 special issues of the International Journal of Multiscale Computational Engineering, organized 5 international symposia on Simulation of Multiphysics Multiscale Systems http://www.science.uva.nl/∼valeria/SMMS/, served as a program committee member and a reviewer in over 20 conferences and 6 international journals, participated in more than 40 conferences, and worked in about 20 international projects. Her research interests include parallel distributed computing in heterogeneous systems, Grid computing, problem solving environments; modeling, simulation and numerical methods in physics.

Prof. Peter M.A. Sloot is a full professor in Computational Sciences at the University of Amsterdam, the Netherlands. He has been the General Chair of the ICCS series of conferences on Computational Sciences since 2002. Dr. Sloot is the Editor in Chief of Elsevier's science journal: Future Generation of Computing Systems and Associate editor of The International Transactions on Systems Science and Applications. More information: http://www.science.uva.nl/∼sloot

Please cite this article as: V.V. Korkhov, et al., A Grid-based Virtual Reactor: Parallel performance and adaptive load balancing, J. Parallel Distrib. Comput. (2007), doi: 10.1016/j.jpdc.2007.08.010