Hierarchical resource management in grid computing

Korkhov, V.V.

Publication date
2009
Document Version
Final published version

Chapter 4. Parallel applications in multi-cluster environment: speedup and efficiency on the Grid

4.1 Introduction

In this chapter we continue to study the multi-layer hierarchy described in Chapter 1. We evaluate the possibilities of running parallel applications in a loosely coupled distributed environment: here we evaluate the 'horizontal' expansion of the parallel applications on the Task layer (see Figure 1.1). A multi-cluster system is used as a resource, and a single parallel application spans over several sub-clusters. We outline the concept of Grid speedup and the theoretical approach for its evaluation introduced in [49] and perform its experimental validation. The Lattice Boltzmann Method (LBM) solver is used as a case study application; a simple model of this application is used for prediction of possible performance gain from the multi-cluster distribution. Compared to the solution for parallel applications spanning over a set of heterogeneous resources proposed in Chapter 3, the approach presented in this chapter is valid for homogeneous Grids, and gives rather high accuracy in predicting the application speedup using a simple application model and a set of basic environment characteristics.

4.2 Speedup and efficiency

In this section we refer to the definitions of speedup and efficiency given in Section 3.4.1 and present more detailed analysis of these metrics.

Degree Of Parallelism (DOP) is defined as the number of processors participating in executing a program at a particular instant over time. That means that when k processors are executing the program, DOP is equal to k. The plot of the DOP over the execution time is usually called Parallelism Profile. The average DOP is defined as the average number of processors used to execute a program.

Let's define the amount of parallel workload (expressed e.g. in Mflop) with a degree of parallelism (DOP) i as W_i [51, 78, 107], the computing capacity of a processor (expressed in e.g. Mflop/s) as Δ. Then the amount of workload executed while running a part of the program with DOP $= i$ is

$$W_i = \Delta i t_i \tag{4.1}$$

where t_i is the total amount of time during which DOP $= i$. The total amount of workload is

$$W = \sum_{i=1}^{m} W_i \tag{4.2}$$

where m is the maximal DOP in the application.

Assuming that the workload W is executed on p processors, the execution time for the portion of work with DOP $= i$ is

$$t_i(p) = \frac{W_i}{i\Delta} \left\lceil \frac{i}{p} \right\rceil \tag{4.3}$$

A possible load imbalance is implicitly taken into account in the formulation of eq. 4.3. The total execution time of workload W on p processors, $T_p(W)$, equals

$$T_p(W) = \sum_{i=1}^{m} t_i(p) = \sum_{i=1}^{m} \frac{W_i}{i\Delta} \left\lceil \frac{i}{p} \right\rceil + Q(W,p) \tag{4.4}$$

with the (communication) overhead for a p processor system for the completion of the workload W defined as $Q(W,p)$, and understanding that $Q(W,1) = 0$.

The *Obtained Speedup* for the workload W on a p processor system is defined as

$$S_p(W) = \frac{T_1(W)}{T_p(W)} = \frac{\sum_{i=1}^{m} W_i}{\sum_{i=1}^{m} \frac{W_i}{i} \left\lceil \frac{i}{p} \right\rceil + \Delta Q(W,p)} \tag{4.5}$$

Amdahl's law and the scaled speedup laws of Gustafson-Baris and Sun-Ni [51] can be derived from these equations.

We will consider first a few simple cases:

1. Assume $W_i = 0$ if $i \neq p$. This means that the application has the only DOP equal to p, thus the application is purely parallel and contains no sequential part. In this case:

$$T_1(W) = T_1(W_p) = \frac{W_p}{\Delta}; T_p(W) = T_p(W_p) = \frac{W_p}{p\Delta} + Q(W_p,p) \tag{4.6}$$

and the expressions for relative speedup and efficiency are:

$$S_p = \frac{T_1(W)}{T_p(W) = \frac{p}{1+f(W_p,p)}}; \varepsilon_p = \frac{T_1(W)}{pT_p(W)} = \frac{1}{1+f(w_p,p)} \tag{4.7}$$

where $f(W,p)$ is the fractional overhead function:

$$f(W_p,p) = p\Delta\frac{Q(W,p)}{W} \tag{4.8}$$

2. Assume $W_i = 0$ if $i \neq 1, i \neq p$. This means that the application consists of the parts with DOP equal to p and to 1, thus the application contains a sequential part and a parallel part with fixed number of processors participating in the parallel execution. In this case:

$$T_1(W) = T_1(W_1 + W_p) = \frac{W_1}{\Delta} + \frac{W_p}{\Delta}; T_p(W_1 + W_p) = \frac{W_1}{\Delta} + \frac{W_p}{p\Delta} + Q(W, p) \quad (4.9)$$

The expression for the speedup now becomes

$$S_p = \frac{T_1(W)}{T_p(W)} = \frac{1}{W_1/W + W_p/pW + f(W,p)/p} = \frac{1}{\alpha + (1-\alpha)/p + f(W,p)/p} \quad (4.10)$$

where $\alpha = W_1/W$, the fraction of sequential workload. Considering $f(W, p) = 0$, equation 4.10 is reduced to Amdahl's law, with the Amdahl bottleneck $1/\alpha$ in the limit of infinite number of processors.

4.3 Parallel applications on a multi-cluster

The core of a theoretical development for the Grid speedup and scaling is given in [49]. In the following sections we will summarize this approach and present an experimental validation.

4.3.1 Hierarchical decomposition of parallel applications

We consider the case of a parallel application with a workload W running on a multi-cluster which is also referred as a Homogeneous Computational Grid (HCG) computing environment: a set of identical parallel systems bound together as a single distributed computing resource. In this case the workload is decomposed among C Computing Elements (CE's), and each CE is a parallel computing system with p nodes that have the same computing capacity Δ. Thus a two-level hierarchical decomposition is introduced: first, the workload is decomposed between C CE's, and then for each CE the portion of workload is again decomposed between the processors of this CE (see Fig. 4.1). Examples of Grid-enabled applications of this kind can be found in literature, e.g. [61].

In Figure 4.1, illustrating the hierarchical decomposition, three levels can be identified. The first level is the full non-decomposed workload W - the level of sequential computing. The next *level 1* introduces the division of the workload between the CE's. This decomposition induces a *level 1* overhead $Q_1(W, C)$, that reflects communication between CE's or load imbalance among CE's. Next, within each CE the

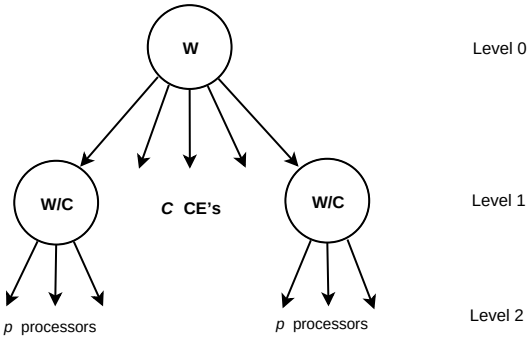

Figure 4.1: Hierarchical decomposition of a parallel application on a homogeneous computational Grid: the total workload W is distributed between C computing elements with p processors each.

workload W/C is again decomposed between the p processors of the CE. On this level another overhead $Q_2(W/C, p)$ is encountered. In case of a single CE (i.e. $C = 1$ and $Q_1(W, 1) = 0$) the standard parallel case is observed where $Q_2(W, p)$ plays the role of $Q(W, p)$ mentioned in the previous section. In homogeneous Grid environment each CE receives an equal share of the total workload. Moreover, the same overhead function $Q_2(W, p)$ applies within each CE.

In the further analysis we consider the case of $W_i = 0$ if $i \neq p$. Let us denote the execution time on the HCG with C CE's and p processors per CE as $T_{C,p}(W)$, and with the definitions introduced earlier we find:

$$T_{C,p}(W) = T_{C,p}(W_p) = \frac{W_p/C}{p\Delta} + Q_2(W_p/C, p) + Q_1(W_p, C) = \frac{W_p}{pC\Delta} + Q_2(W_p/C, p) + Q_1(W_p, C) \quad (4.11)$$

Equation 4.11 points to several facts:

- Running a tightly coupled parallel application decomposed over several CE's in a computational Grid might be of no use because of the large overheads that are be induced by the communication between the CE's (see e.g. [81]). Indeed, this overhead is expressed by $Q_1(W_p, C)$

- On the other hand, the execution of the application on more than one CE attracts more processors for processing (with the assumptions taken, a factor C more processors). Moreover, the overhead per CE is changed, from $Q_2(W_p, p)$ in the case of running on one CE to $Q2(W_p/C, p)$ in the case of running on C CE's.

Adapting the definition of the obtained speedup (eq. 4.5) to the execution in homogeneous multi-cluster environment, we get:

$$S_p^C = \frac{T_{1,1}(W)}{T_{C,p}(W)} \tag{4.12}$$

where a superscript C is added to denote the decomposition over C CE's. The derived expression for the obtained speedup is:

$$S_p^C = \frac{pC}{1 + f_2(W_p/C, p) + f_1(W_p, p, C)} \tag{4.13}$$

where

$$f_1(W, p, C) = pC\Delta\frac{Q_1(W, C)}{W} \tag{4.14}$$

$$f_2(W, p) = p\Delta\frac{Q_2(W, p)}{W} \tag{4.15}$$

Note that $f_2(W, p) = f(W, p)$ (compare eq.4.8 with eq.4.15) and that $f_1(W, p, 1) = 0$. Eq.4.13 reduces to eq. 4.7 for $C = 1$. This shows that the hierarchical decomposition introduces a second fractional overhead f_1 in the expression for the speedup.

4.3.2 Grid speedup

To understand if the obtained speedup is a good metric to asses the added value of decomposing an application over CE's, let us recall the reasons why parallelism was introduced initially:

1. The computational problems were compute bounded: the computing time was unacceptably high on one processor, thus more computing power was needed.

2. The computational problems were memory bounded: the memory consumption of the application was so large that it would not fit in memory of a single processor. Using more computing nodes and distributed memory could increase the amount of available memory for the application.

In order to analyze the added value of parallelism, one compared the execution time of the parallel application with a reference value: the execution time on a sequential computer. In a Grid computing environment the reference value should not be the single processor, but the execution time on one single CE. The reasons to decompose the application over more than one CE are exactly the same as the original reasons to parallelize the application, the computational problem is compute or memory bound in one CE, or a combination of both. So the core question the introduced metric should answer is whether decomposing the application over C CE's gives any added value compared to running it on one CE. This leads us to the concept of Grid speedup, defined as:

$$\Gamma_p^C = \frac{T_{1,p}(W)}{T_{C,p}(W)} \tag{4.16}$$

Grid speedup shows the ratio of the execution time of the application on 1 CE to the execution time on C CE's. Note that Grid speedup depends on two parameters, the number of CE's and the number of processors per CE. In case of $p = 1$ the traditional situation of a parallel computation arises, with C playing the role of the number of processor. Grid efficiency is defined as:

$$\gamma_p^C = \frac{T_{1,p}(W)}{CT_{C,p}(W)} \tag{4.17}$$

Let us compute the Grid speedup for the example of a parallel workload W_p. Substitute eq. 4.11 into eq. 4.13, which after some algebraic transformations results in:

$$\Gamma_p^C = \frac{C}{1 + g_2(W_p, p, C) + g_1(W_p, p, C)} \tag{4.18}$$

Two fractional Grid overhead functions are defined as:

$$g_1(W_p, p, C) = C\frac{Q_1(W_p, C)}{T_{1,p}(W_p)} \tag{4.19}$$

$$g_2(W_p, p, C) = \frac{CQ_2(W_p/C, p) - Q_2(W_p, p)}{T_{1,p}(W_p)} \tag{4.20}$$

The fractional Grid overhead function g_1 plays the same role as the fractional overhead f (Eq. 4.8) in the simple parallel case. Indeed, f can be rewritten as $f = pQ(W_p, p)/T_1(Wp)$ which shows that high Grid speedups can be obtained if the Grid fractional overhead g_1 is small, e.g. the grain size, defined as the portion of work per CE, is large enough such that the amount of work per CE is much larger than the amount of overhead Q_1 induced by the level-1 decomposition.

The hierarchical decomposition introduces another fractional Grid overhead g_2. The peculiarity of this overhead is that in theory it can also be negative, thus improving the Grid speedup due to the subtle changes in the *level 2* overheads when decomposing a parallel application over CE's.

The property of positivity of the Grid overhead g_2 directly depends on the linearity of Q_2 overhead on W_p. Three cases can be distinguished:

1. $aQ_2(W/a, p) = Q_2(W, p)$
2. $aQ_2(W/a, p) > Q_2(W, p), a > 1$
3. $aQ_2(W/a, p) < Q_2(W, p), a > 1$

The first option describes the applications for which the level 2 overheads are proportional to the workload. In this case we see that $g_2 = 0$ and

$$\Gamma_p^C = \frac{C}{1 + g_1(W_p, p, C)} \tag{4.21}$$

For this set of applications good Grid speedups can be obtained if the *level 1* fractional Grid overhead function g_1 is small enough, independent of the quality of the obtained speedup of the parallel application on one CE that is dictated by Q_2. This does not mean that the Grid speedup is independent on p, the number of processors in one CE, but it means that high Grid efficiencies are possible for applications that have a small parallel efficiency on one CE. By keeping the grain size large enough it is theoretically possible to get good benefits by decomposing parallel applications over several CE's.

The second case brings $g_2 > 0$, which has a negative effect on the Grid speedup. The magnitude of g_2 depends on the application details.

The third case shows $g_2 < 0$, which claims a positive effect of a hierarchical decomposition on the Grid speedup. Theoretically this effect can be illustrated by imaging applications for which the overhead of running on one CE is reduced faster then associated reduction of the computational workload by a factor of C. In case such applications exist in practice, they can count on the theoretical possibility for super linear Grid speedups.

4.3.3 Limitations and applicability

Although the theoretical derivations presented in the previous sections can be used to estimate parallel efficiency for real applications, we should mention the practical limitations of the presented approach.

First, the initial assumption about homogeneity of the resources is not often met in existing Grid testbeds. The speed of processors in participating clusters, network bandwidth within and between the clusters can be different which should result in a proper workload distribution between the participating subclusters. Taking a look at figure 4.1 we can see that in case of such heterogeneity of the environment the *level 1* decomposition of the workload between clusters should not be equal any more. Instead, application of workload balancing methods, in particular AWLB discussed in Chapter 3, should result in appropriate sharing of the workload according to resource capacity. In this case the value for $T_{C,p}(W)$ introduced in equation 4.11 changes to reflect unequal workload distribution between the participating clusters. Considering the weight of each cluster, that reflects its capacity, as w_i ($\sum w_i = 1$) the share of the workload for each cluster k is $W^k = w_k W$. The requirement for simultaneous completion of computations on all the clusters results in $T^k = \frac{W^k}{p\Delta^k} = T^{HET}$ where Δ^k represents the computing capacity of a processor in the cluster k, p - number of processors used in each cluster, T^k - execution time in the cluster k which is equal to T^{HET} for all the clusters. Equation 4.11 is transformed to the following:

$$T_{C,p}^{HET}(W) = T^{HET} + Q_2 + Q_1 \tag{4.22}$$

where Q_1 and Q_2 should be also re-calculated according to the workload distribution. The initial idea of the Grid speedup has to be updated in this context, as the single

reference value of the time taken to execute the application on p processors will be different for each cluster because of resource heterogeneity. For example, the shortest time of execution on a single cluster within the cluster set can be used instead.

Second, the assumption about equal number of processors p acquired from each cluster is not often feasible in practice. This leads to the concept of so called "fractional C" that allows to analyze the case in which CE's have different number of processors.

4.4 Case study: Lattice Boltzmann Method solver on DAS-2

4.4.1 Strip wise workload decomposition on a homogeneous multi-cluster

To evaluate the theoretical approach described in the previous section we selected a computational hemodynamic solver HemoSolve [8] that uses Lattice Boltzmann Method (LBM) - a mesoscopic method for fluid flow simulation based on a discretized Boltzmann equation with simplified collision operator [105]. For the experiments we used a basic geometry of the tube with a fluid flow inside (originally simulation of a blood vessel and a blood flow) with circle section (Figure 4.2). The method and solver implementation allow parallelization in such a way that only the exchange of boundary information is needed between the processors. The application can be decomposed between several computational clusters (or Computing Elements, CE), and such decomposition implies that a single processor in one CE exchanges boundary information with another processor in another CE, while at the same time inter CE communication between other processors is executed. In this way we get a form of latency hiding, while intra CE communication takes place, we also perform inter CE communication. We expect that this latency hiding will have a beneficial effect on the overall Grid efficiency.

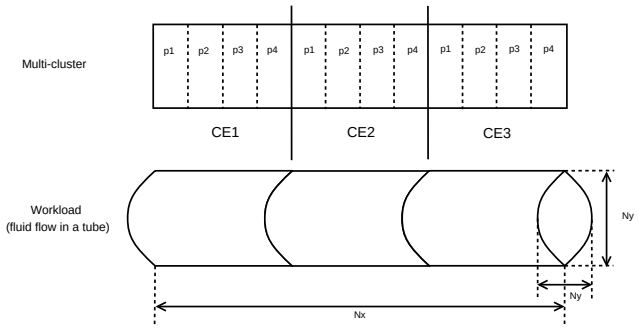

Figure 4.2: Stripwise decomposition of the workload between CE's in the blood flow simulation.

All the experiments were carried out in the DAS-2 multi-cluster environment. It consists of 5 clusters located at different universities in the Netherlands. Each cluster contains at least 32 nodes with the following characteristics: Two 1-GHz Pentium-IIIs; at least 1 GB RAM; A 20 GByte local IDE disk (80 GB for Leiden and UvA); A Myrinet interface card; A Fast Ethernet interface (on-board).

The nodes within a local cluster are connected by a Myrinet-2000 network, which is used as high-speed interconnect, mapped into user-space. In addition, Fast Ethernet is used as OS network (file transport). The five local clusters are connected by the Dutch university Internet backbone [130].

For the experiments a number of tube geometry configurations have been selected:
$N_x = \{36, 64, 128, 256, 512\}$ $N_y = N_z = \{10, 20, 30, 40, 50\}$

where N_x is the length of the tube, $N_y = N_z$ are the diameter of the tube (measured in the points of discretization per dimension).

The theoretical estimation of execution time for LBE code on single processor:

$$T_{C=1}^{p=1} = \frac{N_x N_y^2}{\Delta} \tag{4.23}$$

where Δ is the number of Lattice Updates per second (LUP/s). The execution time on a single CE in this case becomes:

$$T_{C=1}^{p} = \frac{N_x N_y^2}{p\Delta} + T_{comm} \tag{4.24}$$

where T_{comm} is the communication time needed to send the stencil information of a boundary of the processor domain to a neighboring processor.

Let's consider the execution times in a single and multi-cluster environments and derive the value for T_{comm} in eq. 4.24. For a single cluster we get:

$$T_{C=1}^{p} = \frac{N_x N_y^2}{p\Delta} + T_{comm} = \frac{N_x N_y^2}{p\Delta} + 2N_y\tau_{comm} \tag{4.25}$$

where τ_{comm} is the is the communication time needed to send the information of a point on the boundary of the processor domain to a neighboring processor in the cluster. The factor 2 emerges because each processor should communicate with its left and right neighbor. On more than one CE the hierarchical decomposition results in the following execution time:

$$T_{C}^{p} = \frac{N_x N_y^2}{pC\Delta} + N_y^2(\tau_{comm} + \tau_{grid}) \tag{4.26}$$

where τ_{grid} is the time to send the single point information between the border nodes located in different CE's. Note that we assume here a communication pattern where we first communicate between CE's.

4.4.2 Estimation of infrastructure parameters

Based on the experimental data it is possible to evaluate actual values for $\Delta, \tau_{comm}, \tau_{grid}$ in the current experimental environment. From the eq. 4.25 we can derive the expression for Δ:

$$\Delta = \frac{N_x N_y^2}{T_{C=1}^{p=1}} \tag{4.27}$$

Test results with different values of N_x and N_y showed that $\Delta \approx 1.3 \times 10^5 LUP/s$. Further on, we derive and evaluate the value for τ_{comm}:

$$\tau_{comm} = \frac{T_{C=1}^{p} - \frac{N_x N_y^2}{p\Delta}}{2N_y} = \frac{T_{C=1}^{p} p\Delta - N_x N_y^2}{2N_y p\Delta} \tag{4.28}$$

This gives the estimation $\tau_{comm} \approx 5 \times 10^{-6} s$.
The expression for τ_{grid} can be derived in the following way:

$$\tau_{grid} = \frac{T_C^p}{N_y^2} - \frac{N_x}{pC\Delta} - \tau_{comm} \tag{4.29}$$

or

$$\begin{cases} T_{C_1}^{p_1} = \dfrac{N_x N_y^2}{p_1 C_1 \Delta} + N_y^2(\tau_{comm} + \tau_{grid}) \\ T_{C=1}^{p=p_1 C_1} = \dfrac{N_x N_y^2}{p_1 C_1 \Delta} + 2N_y \tau_{comm} \end{cases} \Rightarrow \tau_{grid} - \tau_{comm} = \frac{T_{C_1}^{p_1} - T_{C=1}^{p=p_1 C_1}}{N_y^2} \tag{4.30}$$

The experiments result in the value $\tau_{grid} \approx 3 \times 10^{-5} s$.

4.4.3 Execution time

The core metrics used in the evaluation of application execution in a distributed environment is the execution time. In this subsection we will take a look at typical execution time trends of LBM solver in the DAS-2 testbed depending on application parameters, and resources selected to be used.

Execution time: theoretical and experimental

Now that the estimations for execution environment parameters are available, we can evaluate the theoretical predictions with experimental execution data. Based on the eq. 4.26 we derive the prediction for the execution time and compare it to the results obtained from actual application runs (figure 4.3). We can see the model gives relatively high accuracy in predicting the application runtime: the deviation between the predicted and measured times is less than 5 percent.

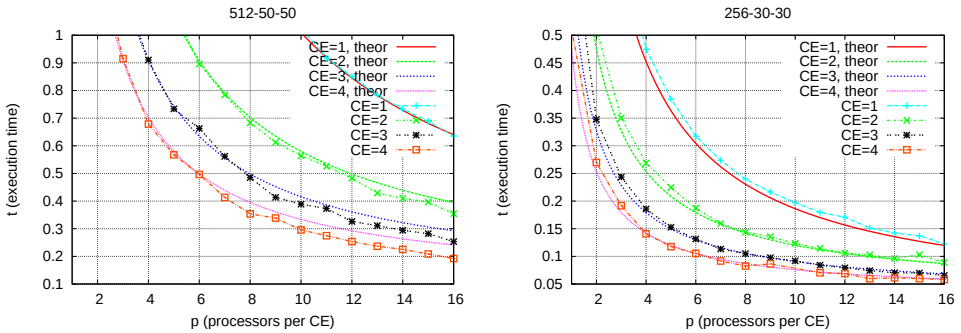

Figure 4.3: Comparison of theoretical prediction vs experimental execution time

Execution time/grain size

One of the important characteristics of the parallel application is the grain size – the size of workload portion per single processor. We define the grain size for the use case application as follows:

$$G = \frac{N_x}{pC} \tag{4.31}$$

Consider the theoretical dependency of the runtime on the grain size. Substituting G into the formula 4.26 we get:

$$T_C^p = G\frac{N_y^2}{\Delta} + N_y^2(\tau_{comm} + \tau_{grid}) = A_1 G + A_2 \tag{4.32}$$

where A_1 and A_2 are constant: $A_1 = \frac{N_y^2}{\Delta}$ and $A_2 = N_y^2(\tau_{comm} + \tau_{grid})$ This theoretical derivation predicts that the dependency of the execution time on the grain side should be linear and result in collapsing curves for all the cases with $CE > 1$. The case of $CE = 1$ should give a parallel line shifted lower by $N_y^2\tau_{grid}$.

Figure 4.4 illustrates the experimental dependency of the execution time on the grain size for different numbers of CEs used. A clear linear dependency is observed, with the curves for the case $CE > 1$ shifted along the Y axis - the influence of the inter-CE communication overhead. The plots for different values of tube width are shown, both keeping the clear linear dependency: the theoretical prediction is confirmed by the experiment.

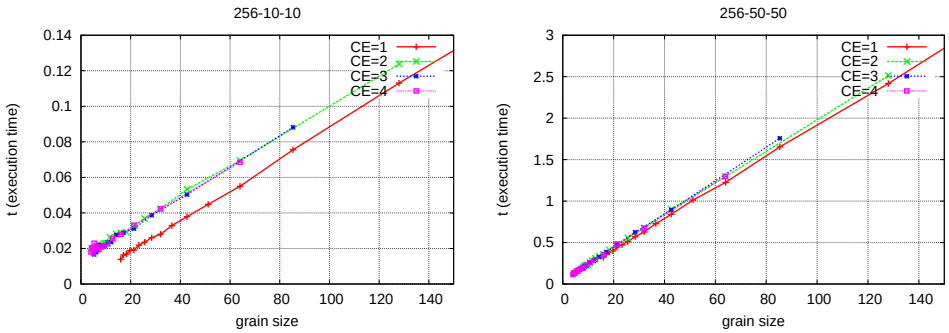

Figure 4.4: Dependency of the execution time on grain size

4.4.4 Grid speedup and efficiency

The main target of this analysis is to check whether the theoretical estimation for the speedup in the distributed multi-cluster environment coincides with the experiment, and if the theoretical expressions can be used to formulate the realistic requirements to the environment to achieve the expected application speedup and efficiency. In this section we examine the behaviour of the Grid speedup and efficiency metric, and compare experimental data with theoretical expectations.

Grid speedup/grain size

The analysis of this dependency gives the insight on the effect introduced by increasing the number of processors to compute the problem of a fixed size, thus decreasing the portion of workload handled by each processor. Naturally, the more processors are involved the less execution time is expected (keeping in mind the limitations put by the Amdahl's law). On the other hand, the larger grain sizes provide higher speedups as the fraction of communications is lower. The application profile of the speedup depending on the grain size gives the instant view on the reasonable size of the workload portions per processor. Figure 4.5 illustrates these dependencies for the tubes of two different sizes. The singularities observed on the left plot are caused by temporary slowdown of the execution for a single measurement in the row of 8 measurements ($\approx$ 5 seconds for the overshoot instead of typical $\approx$ 2 seconds) and do not reflect any application behaviour dependency. This illustrates that even justified assumptions and predictions can not guarantee the performance of a particular execution: distributed environments are typically unpredictable in performance unless special measures are taken towards advance resource and bandwidth reservation, and service-level agreements.

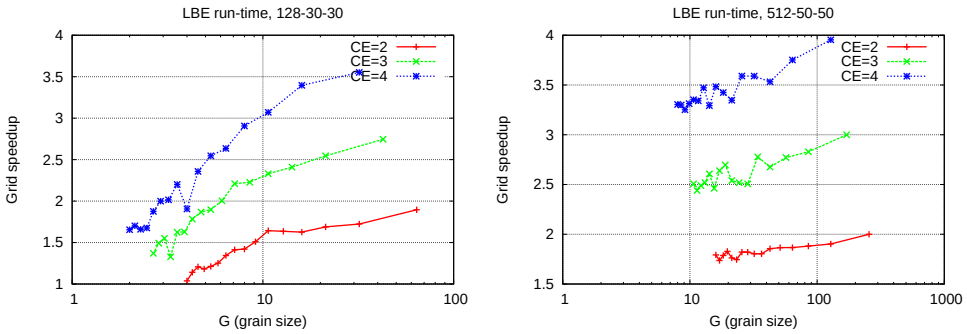

Figure 4.5: Dependency of the Grid speedup on grain size

Grid speedup/processors per Computing Element

In this section we finally address the main metrics that have been introduced to characterize the application execution in the multi-cluster environment. We analyze the theoretical expectations for Grid speedup and efficiency compared to the experimental data, and examine when it pays off to execute the application on more than one CE.

Theoretical expression for the Grid speedup:

$$\begin{cases} \Gamma_C^p = \dfrac{T_{C=1}^p}{T_C^p} = \dfrac{\dfrac{N_x N_y^2}{pC\Delta} + 2N_y\tau_{comm}}{\dfrac{N_x N_y^2}{pC\Delta} + N_y^2(\tau_{comm} + \tau_{grid})} \\ \alpha = \dfrac{\tau_{grid}}{\tau_{comm}} \\ \beta = \dfrac{N_x}{p\Delta\tau_{comm}} \end{cases} \Rightarrow \Gamma_C^p = \frac{C}{1 + \dfrac{C(\alpha+1)-2}{\beta+2}} \quad (4.33)$$

Here the dimensionless parameter α expresses the imbalance in the communication hierarchy between inter- and intra CE communication. The dimensionless parameter β contains the grain size of the application running on one CE, and the balance between computational speed and communication within one CE. Obviously, the Grid speedup increases with larger α and smaller β.

Experimental estimation:

$$\Gamma_C^p = \frac{T_{C=1}^p}{T_C^p} \quad (4.34)$$

The figures 4.6 and 4.7 illustrate the dependencies of the Grid speedup and efficiency on the number of processors used within each CE. We can clearly see that the experimental curves follow the same trends as the theoretical predictions, and the deviation does not exceed 5 percent. These plot can give an instant view on the

reasonable number of processors to use to get a desired value of Grid speedup and efficiency.

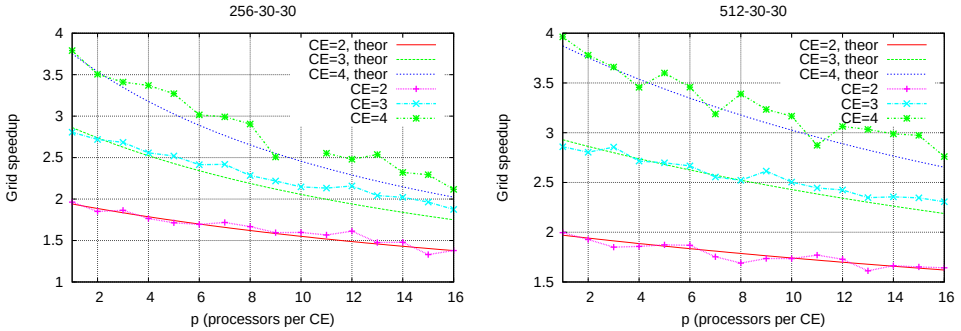

Figure 4.6: Grid speedup

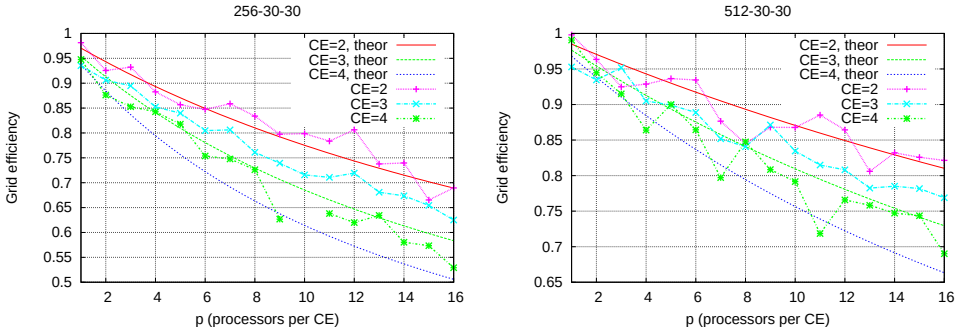

Figure 4.7: Grid efficiency

On some plots a wavy behaviour of the curves is observed. Additional analysis of the experimental data (8 measurements per each point) showed that in some series significant overshoot is observed: one or two measurements of eight in a row differ by up to 30 percent compared to the average value, which results in significant increase of the standard deviation for these series. As the experimental system used for the experiments (DAS-2) is a real Grid-like environment with shared utilization, we explain this deviation by influence of other resource demanding jobs in the system.

Grid speedup/β

A more generic view on the Grid speedup is given while building the dependency on the dimensionless parameter β (see eq. 4.33). In this case a single theoretical profile

fits all the initial data configurations. The figures 4.8 and 4.9 illustrate the Grid speedup and efficiency dependency on β for different tube sizes.

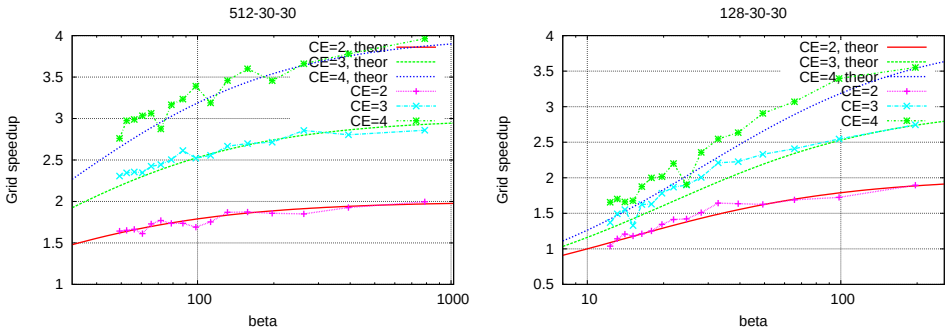

Figure 4.8: Dependency of Grid speedup on β

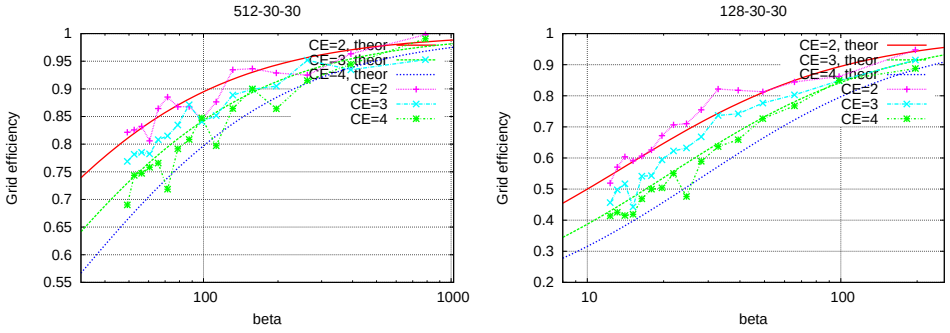

Figure 4.9: Dependency of Grid efficiency on β

The single profile allows building a single dependency of the required β to achieve the desired efficiency.

Consider the requirement $\gamma > \gamma_0$, hence from the equation 4.33 follows:

$$\beta > \frac{\gamma_0(C(\alpha+1)-2)}{1-\gamma_0} - 2$$

Substitution the values obtained in section 4.4.2 gives the value:

$$\alpha = \frac{\tau_{grid}}{\tau_{comm}} = 6.$$

Thus the expression for β becomes:

$$\beta > \frac{\gamma_0(7C-2)}{1-\gamma_0} - 2 \tag{4.35}$$

Consider $\gamma_0 = 0.8$, then

$$\beta^{\gamma>0.8} > 28C - 10 \tag{4.36}$$

Thus $\beta_{CE=2}^{\gamma>0.8} > 46, \beta_{CE=3}^{\gamma>0.8} > 74, \beta_{CE=4}^{\gamma>0.8} > 102$. From the equations 4.33 it follows that $\frac{N_x}{p} = \beta\Delta\tau_{comm}$ or $N_x = \beta\Delta\tau_{comm}p$. Substituting the values we get:

$$CE = 2: \quad N_x = \beta_{CE=2}^{\gamma>0.8}\Delta\tau_{comm}p = 29.9p \tag{4.37}$$

$$CE = 3: \quad N_x = \beta_{CE=3}^{\gamma>0.8}\Delta\tau_{comm}p = 48.1p \tag{4.38}$$

$$CE = 4: \quad N_x = \beta_{CE=4}^{\gamma>0.8}\Delta\tau_{comm}p = 66.3p \tag{4.39}$$

This analysis based on a simple model gives a quick but rather accurate estimation of the pay off to expect from the application running in an environment with known basic characteristics. We can see that the theoretical numbers for reasonable N_x are realistic to be used in practice which is confirmed by the experimental results.

4.5 Conclusions

We have evaluated the Grid speedup metric introduced in [49] that allows analyzing the performance of tightly coupled parallel applications on a Homogeneous Computational Grid. Using the concept of a two level hierarchical decomposition of the workload, a general formalism was introduced that allows computing Grid speedup in terms of two fractional overhead functions. Using this formalism we analyzed a prototypical application, a model for Lattice Bolzmann Method solver and compared the predictions derived from the theoretical approach with the experimental data obtained in a DAS2 multi-cluster environment.

The experiments were used to validate the theory proposed in [49], and they show that the theory can be used indeed for a quick but rather accurate estimation of the pay off to expect from the application running in an environment with known basic characteristics, or what system requirements have to be fulfilled to achieve the required execution efficiency. The limitation of the presented theoretical model is that only homogeneous resources are considered and the number of processors acquired from each cluster is equal for all the participating clusters. These restrictions can be overcome if workload balancing is introduced to ensure proper sharing of the workload in the heterogeneous case, and "fractional C" concept is considered while analyzing the distribution in the environment with different number of available processors per cluster.

In the next chapter we go beyond a single parallel application on a set of distributed resources. Instead of tightly coupled parallel applications discussed in this and previous chapters the attention is focused on a loosely coupled set of tasks assigned to process shared workload. Like in Chapter 3, the main concern is to address the heterogeneity of the environment and study the ways to balance the workload with the help of user-level scheduling.