APPLICATION DEDICATED CLUSTERING

RALF GRUBER, ALESSANDRO DE VITA, MASSIMILIANO STENGEL AND TRACH MINH-TRAN, EPFL

L'adaptation de grappes aux besoins des applications est démontrée utilisant un programme en production de type Car-Parrinello. Pour ce faire, l'application et la grappe de processeurs sont parametrisées ce qui donne la possibilité d'une prédiction a priori *du système de communication de la grappe. Une comparaison entre les performances prédites et mesurées sur Swiss-T1 valide la théorie.*

By means of a first principles chemistry Car-Parrinello program, it is shown that commodity clusters can be tailored to applications. For this purpose, a parameterisation of the applications and the parallel machines is made that enables an a-priori prediction of the application behaviour on a given cluster, or, inversely, of an application-dedicated cluster design. Comparisons between predicted and measured performance results on the Swiss-T1 cluster support the theory.

INTRODUCTION

The effective performance achieved on a cluster depends on different parameters. The individual processor is characterised by his cycle time, by the number of results per cycle, by the cache size, by the local memory size, and by the memory bandwidth. The communication system is characterised by the bandwidth and the latency of one link that interconnects two processors and by the network topology.

Up to now, an engineer had to adapt his application to a specific computer architecture. Today, it is possible to tailor computer configurations to the needs of applications. In fact, processing units and network communication systems can be purchased separately. With the standard PCI ports, the standard MPI (Message Passing Interface) communication library, and the resource management systems, it is possible to build vendor-independent, cost-effective parallel machines. First one chooses the best suited computer on which the algorithms are optimised. Knowing the behaviour of the application program on a computer and its communication needs, it is possible to predict the bandwidth/latency of the most suited communication system that interconnects the processors. A multi-machine resource management system together with the MPI library enable heterogeneous configurations. Thus, a commodity cluster can evolve in time with the most recent processors.

The choice of the ideal cluster can be based on the parameterisation of the applications and the parallel machine [11]. The major application parameters are its effective processor peak performance r_a, and the γ_a quotient of the number of operations and the number of data to be transfered to the other processors. The parallel machine is characterised by the γ_m quotient of r_a and the effective total inter-processor communication bandwidth. If γ_m is smaller than γ_a, the cluster is well tailored to the application. We shall see that γ_m should rather be 4 times smaller than γ_a such that communication does not take more than 20% of the overall execution time.

This concept has been tested with the first principles chemistry Car-Parrinello program LAUTREC [1] for which an a-priori estimation has been made for these parameters and validated by benchmark measurements made on the Swiss-T1 Alpha cluster [2, 3].

APPLICATION RELEVANT PARAMETERS

The parallel application can be parameterised by the quantity

$$\gamma_a = \frac{\text{number of operations [Mflops]}}{\text{amount of data to transfer [Mwords]}} = \frac{O}{S} \qquad (1)$$

The number of operations O corresponds to the computational effort that has to be made on a single processor and S denotes the amount of data that has to be transfered from one processor to the other processors. These quantities can be estimated for each application or algorithm.

The computing time needed to compute the O operations on an individual processor depends on the implementation of algorithms on the computer architecture. Each algorithm of the application can be modeled by

$$r_a = min\,(R_\infty\,,\, M_\infty * V_a) \qquad (2)$$

In this formula, the processor relevant quantities R_∞ (in Mflops/s)[1], and M_∞ (in Mwords/s)[2] denote the theoretical peak performance of the processor, and the maximum memory bandwidth, respectively. The quotient

$$V_m = R_\infty / M_\infty \qquad (3)$$

varies from 4 for a Pentium 4/1.6 GHz with fast Rambus memory to 21 for an AMD Athlon/1.4 GHz with slow SDRAM memory.

The application relevant quantity V_a is given by

$$V_a = F_a / W_a \qquad (4)$$

1 Mflops/s=million floating point operations per second
2 Mwords/s=million 64 bit words (or operands) per second

where F_a (in Mflops) and W_a (in Mwords) are the number of operations and the number of main memory accesses, respectively. In the SAXPY operation $\mathbf{y} = \mathbf{y} + a * \mathbf{x}$, in which **x** and **y** are considered to be long vectors of length n and the quantity a is a constant, the number of fundamental vector operations (one multiply and one add) is $F_a = 2n$, and the number of loads (load of vectors **x** and **y** from main memory) and stores (store **y** back to main memory) is $W_a = 3n$. Thus, $V_a = 2/3$ and the expected peak performance is limited by two thirds of the peak memory bandwidth in words per second. Typically, BLAS1 and BLAS2 operations are limited by memory bandwidth and not by processor performance. On the other hand, BLAS3 operations that make up the LINPACK best effort benchmark [4, 5], have big V_a values and the effective peak performance is $r_a = R_\infty$.

It is advised to minimise the communication needs or to maximise γ_a. Thus, the demands on the network communication bandwidth can be lowered and the costs reduced. An additional recommendation is to minimise the number of messages sent such that the latency in the communication network does not impair the communication speed. Both demands can be satisfied by choosing the standard communication library MPI, designed to pass messages between processors. In a cost-effective commodity cluster computer, there is no single image operating system that supports data handling. As a consequence, the programmer has to take care of the data organisation and of the data transfer over the network. This is in contrast to a NUMA server where the system considers the distributed memory as virtually shared. This facilitates the use of parallel machines, the data has not to be distributed by the programmer. In such machines, the overall efficiency of an application can suffer and the number of virtual shared memory processors is limited. If this limit has to be overcome, a NUMA clustering has to be made and the users have to switch to MPI programming. It is therefore highly recommended to design parallel applications for distributed memory cluster architectures and use the standard MPI communication library.

There are three major types of parallel applications; those dominated by point-to-point communications, those with multicast such as all-to-all communications, and those called *embarassingly parallel* applications based on a master-slave concept. The parallel finite element method based on domain decomposition is an example of the first local neighbor communication type. For those applications, the number of processors grows with the size of the problem, the local granularity being fixed. This makes γ_m independent of the number of processors and scalability with the number of processors can be guaranteed. An example of the second type of problems, in which the communication effort is dominated by multicast message transfers of the 3D FFT algorithm, is discussed in detail in this paper. In the *embarassingly parallel* applications, there are no inter-slave communications, but the master node has to distribute the tasks and collect the results. For each of those applications, a special master-slave interconnection concept has to be implemented. Examples of master-slave applications are database, data mining, Web transactions, or sequencing algorithms in genomics and proteomics.

PARAMETERISATION OF CLUSTER COMPUTERS

A cluster can be characterised by:

$$\gamma_m = \frac{\text{effective processor performance [Mflops/s]}}{\text{network bandwidth per processor [Mwords/s]}} = \frac{r_a}{b} \qquad (5)$$

This quotient γ_m describes how many operations can be performed on a processor during the time needed to send one word of data from one processor to another one. Assume that we have P processors interconnected by a network characterised by a total effective communication bandwidth of C Mwords/s and an average distance between the network nodes of $\langle d \rangle$. The effective bandwidth per processor then is

$$b = \frac{C}{P\langle d \rangle} \quad \text{Mwords/s} \qquad (6)$$

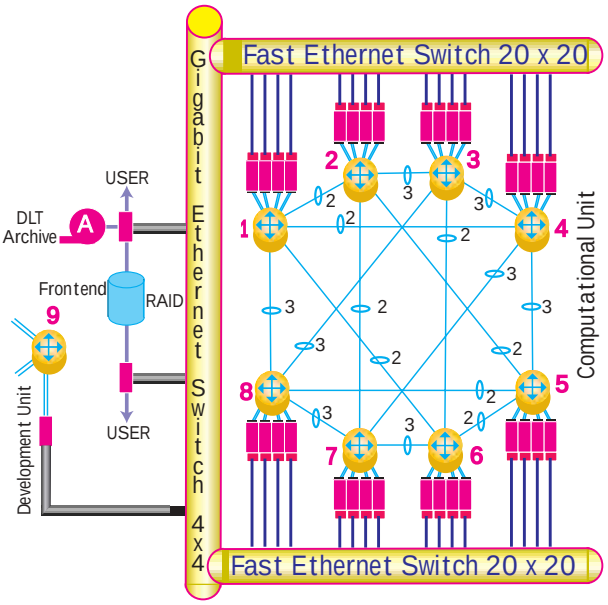

Fig. 1 – The architecture of the Swiss-T1 with its eight 12x12 crossbars and the Fast Ethernet switch connections. The numbers on the links of the Circulant Graph network architecture denote the number of data transfers for an all-to-all operation between the crossbars.

Let us consider the computational unit of the Swiss-T1 machine [3] (see Fig. 1) consisting of a frontend and a computational unit with 32 dual Alpha processor Compaq DS20E computers. Each of the eight 12x12 crossbar switches of its TNet network connects to 8 processors, four links are used to interconnect the

crossbars. Each link reaches 160 Mbyte/s = 20 Mwords/s bidirectional communication bandwidth if one admits double precision 64 bit arithmetics. Since there is no total connectivity in the TNet network, the messages have sometimes to pass over an intermediate crossbar used as a hub. This results in an increased load of the network. In fact, in the circulent graph network configuration used in the Swiss-T1 machine, the 8 crossbars, each one forming a group of 8 connected processors, have different distances. The distances between crossbars can be 0, 1 or 2. For crossbar 1 in Fig. 1, the 8 processors that are directly connected to this crossbar have a distance of 0, the processors connected to the crossbars 2, 4, 6 and 8 have a distance of 1 and, finally, those processors connected to the crossbars 3, 5 and 7 have a distance of 2. The average distance between crossbars then is:

$$\langle d \rangle = (1 \times 0 + 4 \times 1 + 3 \times 2) / 8 = 1.25 \qquad (7)$$

The total number of links between the crossbar switches is equal to $L = 16$, the total uni-directional network bandwidth is $C = 640$ Mwords/s, and the average network bandwidth per processor is $b = 640 / (1.25 \times 64) = 4$ Mwords/s, or 32 MB/s. This bandwidth can only be reached when there is no bottleneck in the network, i.e. if the network is well equilibrated. The communication bandwidth between groups of 8 processors is given by the bandwidth of a PCI port that has been measured to be 80 MB/s, or 10 Mwords/s. When sending a message between two processors of two neighboring groups, two PCI ports and one inter-crossbar link are activated. This leads to a well equilibrated network.

TAILOR CLUSTERS TO APPLICATIONS

We remind that a given application is characterised by γ_a (eq.1) and by its local processor performance r_a (eq. 2). A parallel machine that is well adapted to such an application should have a $\gamma_m < \gamma_a$ or

$$\Gamma = \gamma_a / \gamma_m > 1 \qquad (8)$$

Then, communications do not dominate computations. However, for commodity components for which communication time cannot be hidden behind computation, it is advisable to have a γ_m value that is four times smaller than γ_a. Then, the communication time takes not more than 20% of the overall time.

For point-to-point communication dominated algorithms, such as those derived from finite element methods, γ_a can be high, and the requested bandwidth of the communication network is small. As a consequence, Fast Ethernet switch connections can provide sufficient bandwidth. For communication intensive algorithms, such as the FFT, the communication bandwidth has to be high, and the communication must be based on high performance networks like Myrinet [6], Quadrics [7], or TNet [8, 9].

THE LAUTREC [1] TEST APPLICATION

First principles chemistry applications tackle a class of fundamentally *entangled* problems, which can be highly communication-intensive due to the underlying formalism. For such a quantum chemistry Car-Parrinello application [10], the γ_a factor is estimated, using a model momentum space algorithm in which each electron state is expanded in a basis of plane waves. They are easily evaluated by passing from real to reciprocal space by means of (Fast) Fourier Transformations (FFT). Energy minimisation algorithms are iterative, and basically involve BLAS3 dense linear algebra operations besides the mentioned FFTs. We assume the parallelisation strategy to be based on distributing the FFT coefficients across the processing elements.

NUMBER OF OPERATIONS

For our analysis, we assume that most of the CPU workload is due to

a) scalar products between wavefunctions (i.e. matrix-matrix multiplications, indicated by MM);
b) FFT transforms of the wavefunctions (indicated by FFT).

The higher the number of orbitals and mesh cells, i.e. the bigger the problem, the better this assumption is satisfied.

A detailed study on the number of operations needed for a) and b) can be found in [11]. In this paper, we only present the relevant results for the test case using $N_{orb} = 128$ orbitals and a three dimensional domain cut in $V = 96^3$ cells. This is a medium size system with 32 water molecules. For this case, $V_a = N_{orb}/2$ for MM and $V_a = 8$ for FFT. For an Alpha processor of the Swiss-T1 machine, the major formulas write:

$$\gamma_a = \frac{5}{3} * \frac{P}{P-1} * (0.65 * N_{orb} + 4.24 * \log_2 V) \qquad (9)$$

$$r_a = R_\infty \qquad (10)$$

$$\gamma_m = \frac{r_a}{b} \qquad (11)$$

We realise that γ_a grows linearly with the number of orbitals and logarithmically with the number of cells.

RESULTS USING TNET ON THE SWISS-T1

For the Swiss-T1 cluster using the TNet as communication system, $R_\infty = 1000$ Mflops/s, $b = 10$ Mwords/s, thus $\gamma_m = 100$. For the upper mentioned H_2O test case with $N_{orb} = 128$ and $V = 96^3$, $P = 8$ processors have been used. As a consequence, $\gamma_a = 330$ and $\Gamma = 3.3$.

The execution time has been evaluated analytically to 20.4 seconds and the communication time to 7.5 seconds. This gives a communication over computation

quotient of c_{anal}=0.37. These estimations are based on r_a and b. Measurements on the Swiss-T1 give 36.6 seconds for computing and 11 seconds for communication, leading to γ_m =3.6, and to a communication over computation quotient of c_{meas} = 0.3. These values are close to the analytic estimations.

The Swiss-T1 cluster has no special I/O processors as the Cray T3E. The message passing communication has to be performed by the processors, and the communication time T_{send} has to be added to the computing time T_{comp}. The total time T_{meas} then is

$$T_{meas} = T_{comp} + T_{send} = (1 + c_{meas}) * T_{comp} \qquad (12)$$

One sees that the communication takes $c_{meas}/(1+c_{meas})$ fraction of the total excution time. For our test case, this corresponds to 25%. It implies that the TNet is well suited to the LAUTREC application.

Results using Fast Ethernet on the Swiss-T1

The only parameters that change with respect to the TNet are the bandwidth per link that is now 12.5 MB/s, or b=1.5 Mwords/s instead of 10 Mwords/s. In addition, there is only one link per dual processor computer instead of the 2 links for TNet. Thus, the bandwidth per processor is 13 times smaller, and γ_m = 2000/1.5 = 1300. The message passing interface is MPICH from Argonne. The total time T_{meas} for the Fast Ethernet can be estimated to

$$T_{meas} = (1 + \frac{\gamma_m}{\gamma_a}) * T_{comp} = 5 * T_{comp} \qquad (13)$$

Using the 8 processors of four DS20E machines with a Fast Ethernet communication switch, the total estimated execution time is dominated by communication that takes 80% of the total time, and only 20% are due to computation.

Measurements with LAUTREC made on the Swiss-T1 with Fast Ethernet connections confirm these estimations.

Influence of latency

In this study, the influence of the latency has not been taken into account. In fact, the MPI latency of the TNet is 13 μs that corresponds to up to 13'000 arithmetic operations or to the communication time of a message with one hundred 64 bit words. In the first principles chemistry applications using 3D-FFT transformations, the average message length is $V * \frac{\pi}{16} / P^2$. For the test case with 96 cells, the message length is 40'000 Bytes, or 5'000 words, thus is much bigger than 100 words, and latency does not play a role in the overall communication time. The same is true for Fast Ethernet communication.

Conclusions

The parameterisations of the applications by γ_a and r_a and of the parallel machines by γ_m open the path versus a tailoring of the clusters to the applications. Such dedicated machines are cost-effective and can be upgraded with the most recent processors.

The presented theory has been validated with the LAUTREC test application running on the Swiss-T1 cluster over the fast TNet network and over the Fast Ethernet switch.

It has been shown that the Swiss-T1 cluster is well adapted to the Car-Parrinello application when running over the TNet communication system.

This intensive study is the basis to further design application dedicated clusters based on low-cost components from the mass market.

Acknowledgements

The Swiss-Tx project has been financed by the CTI (Commission for Innovation and Technology). A special acknowledgement goes to all of the over 80 persons who contributed to this project, some with small parts of their time, others full-time. This whole Swiss-Tx team contributed with their skills to this paper. Thanks go also to all our American friends who meet with us once per year at the Commodity Computing workshops [12].

References

[1] A. De Vita, A. Canning, G. Galli, F. Gygi, F. Mauri, and R. Car, *Quantum molecular dynamics on massively parallel computers*, EPFL Supercomputing Review **6** (1994)

[2] R. Gruber, and A. Gunzinger, *The Swiss-Tx Supercomputer Project*, Speedup 11 (1997) Number 2, p.20-26, *http://capawww.epfl.ch/swiss-tx and http://tone.epfl.ch*

[3] P. Kuonen, and R. Gruber, *Parallel Computer Architectures for Commodity Computing*, EPFL Supercomputing Review **11** (1999) 3-11

[4] *http://www.netlib.org/scalapack*

[5] *http://www.top500.org*

[6] *http://www.myri.com*

[7] *http://www.quadrics.com*

[8] S. Brauss, M.Frey, A.Gunzinger, M.Lienhard, and J. Nemecek, *Swiss-Tx Communication Libraries*, Proc. of HPCN99 (Amsterdam, 12-14 April, 1999), 10

[9] Stephan Brauss, *Communication libraries for the Swiss-Tx machines*, EPFL Supercomputing Review **11** (1999) 12-15

[10] R. Car and M. Parrinello, Phys. Rev. Lett. **55**, 2471 (1985)

[11] R. Gruber, P. Volgers, A. DeVita, M. Stengel, and T. M. Tran, *Parameterisation to tailor commodity clusters to applications*, accepted to appear in J. Future Generation Computer Systems (Elsevier, 2002)

[12] *http://www.cs.sandia.gov/Conferences/SOS* ■