COMPLEX PERFORMANCE EVALUATION OF PARALLEL LAPLACE EQUATION

[a] PETER HANULIAK

Polytechnic institute, Sladkovicova 533/20, 018 41 Dubnica nad Vahom, Slovakia
email: [a]phanuliak@gmail.com

Abstract: With the availability of powerful personal computers and networking devices, the recent trend in parallel computing is to connect a number of individual workstations (PC, PC SMP) to solve computation-intensive tasks in parallel way on such clusters (NOW, SMP, Grid). Current trends in high performance computing (HPC) are to use networks of workstations (NOW, SMP) as a cheaper alternative to traditionally used massively parallel multiprocessors or supercomputers. In this article we discuss such complex performance evaluation of iterative parallel algorithms (IPA) and their practical implementations (Jacobi and Gauss-Seidel iteration). On real example we demonstrate the influences in process of modelling and performance evaluation and the consequences of their distributed parallel implementations.

Keywords: network of workstations, iterative parallel algorithms, decomposition strategy, inter – process communication, performance evaluation

1 Introduction

There has been an increasing interest in the use of networks of distributed workstations (cluster) connected together by high-speed networks for solving large computation-intensive problems. Network of workstations (NOW) [7, 10] has become a widely accepted form of high-performance parallel computing. Each workstation in a NOW is treated similarly to a processing element in a multiprocessor system. However, workstations are far more powerful and flexible than processing elements in conventional multiprocessors.

2 Models of parallel systems

In principal we can divide parallel algorithms into two following classes

- parallel algorithm using shared memory. These algorithms are developed for parallel computers with dominated shared memory as actual symmetrical multiprocessors or multicore systems (SMP)

- parallel algorithm using distributed memory (DPA). These algo-rithms are developed for parallel computers with distributed memory as actual NOW system and higher integration Grid systems.

The main difference is in form of inter - process communication (IPC) among individual parallel processes

The load balancing, inter process communication and transport protocol for such machines are being widely studied [4, 9, 10]. With the availability of cheap personal computers, workstations and networking devises, the recent trend is to connect a number of such workstations to solve computation intensive tasks in parallel processes.

3 Complex performance evaluation

To performance evaluation of parallel algorithms we can use analytical approach to get under given constraints analytical laws or some other derived analytical relations [2]. The most known analytical relations have been derived without considering architecture and communication complexity. In NOW [4, 6], we have to take into account all aspects that are important for complex performance evaluation. Theoretically we can use following solution methods to get a function of complex performance

- analytical
  - application of queuing theory results [8, 9]
  - order (asymptotic) analyse [5]
  - Petri nets [13]
- simulation methods [1]
- experimental
  - benchmarks [15]
  - direct measuring [12,11].

Analytical method is a very well developed set of techniques which can provide exact solutions, but only for a very restricted class of models. For more general models it is often possible to obtain approximate results significantly more quickly than when using simulation, although the accuracy of these results may be difficult to determine.
Simulation is the most general and versatile means of modelling systems for performance estimation. It has many uses, but its results are usually only approximations to the exact answer and the price of in-creased accuracy is much longer execution times.
Evaluating system performance via experimental measurements is a very useful alternative for computer systems. Measurements can be gathered on existing systems by means of benchmark applications that aim at stressing specific aspects of computers systems.

3.1 Performance evaluation metrics

To evaluating parallel algorithms there have been developed several fundamental concepts.

Speed up

Let O(s, p) be the total number of unit operations performed by p processor system, s defines size of the computational problem and T(s, p) be the execution time in time units. Then speedup factor is defined as

$$S(s,p) = \frac{T(s,1)}{T(s,p)}$$

Efficiency

The system efficiency for a p processor system is defined by

$$E(s,p) = \frac{S(s,p)}{p} = \frac{T(s,1)}{p\,T(s,p)}$$

Isoefficiency concept

We denote the workload w = w (s) as a function of size of the problem s. For parallel algorithms we define an isoefficiency function relating workload to machine size p needed to obtain a fixed efficiency E when implementing a parallel algorithm on a parallel computer. Let h (s, p) be the total overhead function involved in the parallel algorithm implementation. The efficiency of a parallel algorithm is defined as

$$E(s,p) = \frac{w(s)}{w(s) + h(s,p)}$$

The workload w(s) corresponds to useful computations while the overhead function h(s, p) represent useless overheads times. With a fixed problem size, the efficiency decreases as p increase. Therefore, one can expect to maintain a constant efficiency if the workload w is allowed to grow properly with increasing machine size (scalability).
We rewrite equation for efficiency E(s, p) as follows

$$w(s) = \frac{E}{1-E} h(s,p)$$

The factor C = E / 1 - E is a constant for a given fixed efficiency E.

4 System of linear equations

There exist many various ways how to solve system of linear equations. But there does not exist any universal optimal way of solving it. The existed methods can be divided into exact (finite) and iterative ones.

4.1 Exact methods

These methods come after deterministic number of steps to the exact solution. To these methods belong
- Cramer rule
- Gaussian elimination methods (GEM) and their alternatives.

Cramer rule

Application to solving system of linear equations has its bottlenecks in the extensive calculation of sub determinants. If the number of un-known n is high, so the whole computation time for individual sub determinants raise exponentially [4]. This fact does not change the possibility to compute the sub determinants in a parallel way.

Gaussian elimination method

Gaussian elimination method GEM (commonly marked as LU factorisation), which are with their known alternatives (GEM with pilot element, Gauss-Jordan elimination, special types of systems - Choleskyfactorisation etc.) the most used exact methods of solving the system of linear equations. For this methods were developed sequential and parallel versions of application algorithms. To the standards belong BLAS (Basic Linear Algebra Subprograms), and innovated versions LINPACK (Linear Package), LAPACK (Linear Algebra Package), ScaLAPACK (Scalable LAPACK), PBLACS (Parallel Basic Linear Algebra Communication Subprograms).

4.2 Iteration methods

In using the exact methods on the computers we could not come in many cases to the exact solution. To these methods belong iterative methods
- Jacobi iteration method
- Seidel iteration method.

The difference to the Jacobi iteration method is in it that she uses also the newest just computed elements of new calculated iteration vector (in the same iteration step) to the calculation of other elements. This is in matrix form as

$$X^{(k+1)} = D \cdot X^{(k+1)} + H \cdot X^{(k)} + P$$

, where D is lower triangular part M and H is the upper triangular part of matrix M.

5 Parallel applications of iterative algorithms

Partial differential equations (PDE), is the equation involving partial derivates of an unknown function with respect to more than one independent variable [3, 14]. PDEs are of fundamental importance in modeling all types of continuous phenomena in nature. We will confine our attention to PDE with two space independent variables x, y. The needed function we denote as u (x, y). The considered partial derivations we denote as u_{xx}, $u_{xy,}$ u_{yy}etc. For practical use the most important PDE are two ordered equations and that mainly
- heat equation, $u_t = u_{xx}$
- wave equation, $u_{tt} = u_{xx}$
- Laplace equation $u_{xx} + u_{yy} = 0$.

Here we show how to solve in parallel way specific PDE – Laplace equation in two dimensions – by means of a grid computation method (Fig.1.) that employs finite difference method. Although we focus on this specific problem, the same techniques are used for solving other PDE extensive approximations calculations on various parallel computers (supercomputers, NOW, Grid). Laplace equation is a practical example of using iterative methods to its solution. The equation for two dimensions is following

$$\frac{\delta^2 \Phi}{\delta x^2} + \frac{\delta^2 \Phi}{\delta y^2} = 0$$

Fig. 1.Grid approximation of Laplace equation.

Function Φ (x,y) represents some unknown potential, such as heat, stress etc. Given a two-dimensional region and values for points of the region boundaries, the goal is to approximate the steady-state solution Φ (x,y) for points in the interior by the function u (x,y). We can do this by covering the region with a grid of points (Fig. 1) and to obtain the values of u (xi,yj) = u i,j. Let us consider square region (a,b) x (a,b).
For coordinates of grid points is valid
xi = i*h, yj = j*h, h = (b-a) / N for i,j = 0, 1, ..., N.
We replace partial derivations of Φ ~ u(x,y) by the differences of u i,j. After substituting we obtain final iteration formulae as

$$X_{i,j}^{(t+1)} = (X_{i-1,j}^{(t)} + X_{i+1,j}^{(t)} + X_{i,j-1}^{(t)} + X_{i,j+1}^{(t)}) / 4$$

orits alternative version

$$X_{i,j}^{(t+1)} = (4\, X_{i,j}^{(t)} + X_{i-1,j}^{(t)} + X_{i+1,j}^{(t)} + X_{i,j-1}^{(t)} + X_{i,j+1}^{(t)}) / 8$$

Each interior point is initialised to some value. The steady-state values of the interior points are then computed by repeated iterations. In each iteration the new value of a point is set to a combination of the previous values of neighbouring points. The computation terminates either after a given number of iterations or when every new value is within some acceptable difference Epsilon > 0 of the previous value.

5.1 Local communication

For Jacobi finite difference method a two-dimensional grid is repeatedly updated by replacing the value at each point with some function of the values at a small fixed number of neighbouring points. The common approximation structure uses a four-point stencil to update each element Xi,j (Fig. 2.).

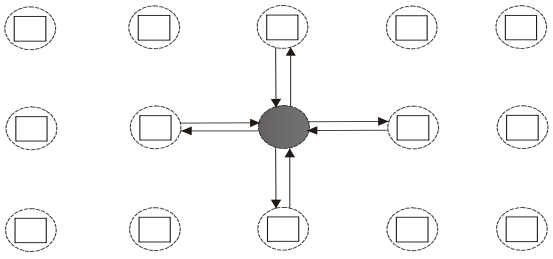

Fig. 2. Communication for 4 - points approximation.

To perform the needed communication we have to do

```
for t = 0 to t–1 do
  begin
   send X_{i,j}^(t) to each neighbours;
   receive X_{i-1,j}^(t),X_{i+1,j}^(t),X_{i,j-1}^(t),X_{i,j+1}^(t) from
neighbours;
   calculate X_{i,j}^(t+1) using derived relations;
  end for;
```

6 Method of direct measuring

The flow diagrams for measurement in SMP parallel computers and for distributed parallel computers (NOW, Grid) differ in form of IPC communication on the level of inter-process communication (IPC) among decomposed parallel processes. The developed iterative parallel algorithms were divided to the two logical parts – manager and worker programs. Both programs are in Win API. Part manager is an application and part service is its service. Manager control the computer with starting services (sending the initial values to all worker computation), makes the connections and starts in parallel way the remote functions. At the end gathers the particular results. Service waits to starting point of the solving in manager controlled interval. After computation it gets back to manager computing results including time of its duration.

7 Decomposition models

To experimental measurement we developed effective iterative parallel algorithms (IPA) to solving Laplace equation with Jacobi methods and Gauss-Seidel (Red-black parallelisation) with following input parameters

- n – number of grid points
- E – accuracy of iteration (Epsilon)
- p – number of processors.

The parallel algorithms were developed in C language using MPI functions in version MPICH (Message Passing Interface Chameleon). The algorithms are divided into two logical parts – manager and worker program. Manager controls every worker through starting services (initial values to all workers), makes the connections and starts the remote functions. At the end manager gathers from worker particular results.
Iteration process ends in reaching defined accuracy or in over exceeding defined number of iterations. After computation service program get back to manager computing results including time of its duration. To measurements we used known decomposition methods (Fig. 3.).

- decomposition of grid points to strips – decomposition 1. At each iteration „boundary points“ (always two boundary rows) are shared by neighbouring processors and after each iteration are interchanged according to Fig. 3.
- decomposition of grid points to square blocks – decomposition 2. In each iteration „boundary points“ (always four edges) are shared by neighbouring processors and the points are interchanged after each iteration along every of these four these edges principally according to Fig. 3

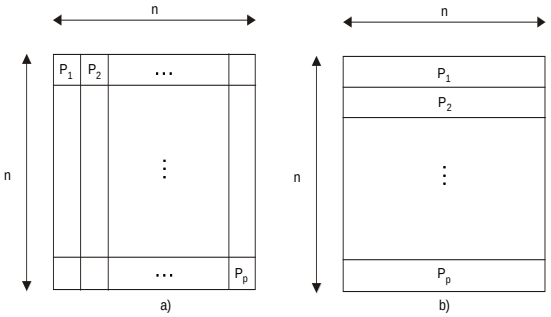

Fig. 3. Decomposition a) blocks b) strips.

8 Results in NOW

All the realised measurements of developed IPA in NOW can be divided to

- calibration of the used workstations in NOW – verifying of their performance for various workload
- measurements of the calculation times and their individual components in relation to workloads for various values of input parameters.

From achieved results I illustrate at Fig. 4. The individual relative parts of whole performance time (Gathering data, Communication time, Calculation time, Initialisation time) for given matrix sizes. To other results I will referee in my next articles

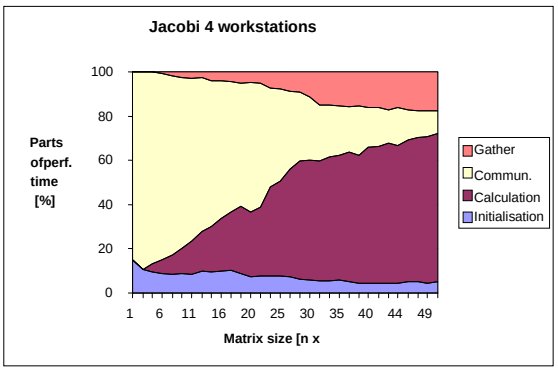

Fig. 4. Individual relative parts of performance time.

9 Conclusions

Therefore in relation to our achieved results we are able to do better load balancing among used network nodes (performance optimisation of parallel algorithm). For these purposes we can use calibration results of individual network nodes in order to divide the input load according the measured performance power of used network nodes. Second we can do load balancing among network nodes based on modern SMP parallel systems and on network nodes with only single processors. Generally we can say that the parallel algorithms or their parts (processes) with more communication (similar to analysed Gauss-Seidel parallel algorithm) will have better speed-up values using modern SMP parallel system as its parallel implementation in NOW. For the algorithms or processes with smaller communication overheads we can use the other network nodes based on single processors.

Literature:

1. Fortier P., Howard M., *Computer system performance evaluation and prediction*, 544 p., 2003, Digital Press
2. Gelenbe E., *Analysis and synthesis of computer systems*, 324 pages, published April 2010, Imperial College Press
3. Došlý O., Rehák P., *Half-linear differential equations*, North Holland, 552 pp., 2005
4. Hanuliak I., Hanuliak P., *Performance evaluation of iterative parallel algorithms*, Kybernetes, Volume 39, No.1, 2010, pp. 107- 126, United Kingdom
5. Hanuliak J., Hanuliak M., *Analytical modelling of distributed computer systems*, NOW, In Proc.: TRANSCOM 2005, pp. 103-110, Žilina
6. Hanuliak I., Hanuliak J., *To performance evaluation of distributed parallel algorithms*, Kybernetes, Volume 34, No. 9/10, West Yorkshire, pp 1633-1650, 2005, United Kingdom
7. Hudik M., *Performance optimization of broadcast collective operation on multi-core cluster*, ICSC Leden 2012, Kunovice, Czech Republic (in print)
8. Hudik M., Hanuliak P., *Parallel complexity of linear system equation*, In Proc.: TRANSCOM 2011 - section 3, pp. 107-110, 2011, Žilina
9. Janovič F., *Performance modelling of distributed parallel algorithms*, ICSC Leden 2012, Kunovice, Czech Republic (in print)
10. Kirk D. B., Hwu W. W., *Programming massively parallel processors*, Morgan Kaufmann, 280 pages, 2010
11. Kumar A., Manjunath D., Kuri J., *Communication Networking*, 750 pp., 2004, Morgan Kaufmann
12. Lilja D. J., *Measuring Computer Performance*, 280 pages, 2005, University of Minnesota, Cambridge University Press, United Kingdom
13. Paterson D. A., Hennessy J. L., *Computer Organisation and Design*, 912 pp., Morgan Kaufmann, 2009
14. Powers D., *Boundary value problems and partial differential equations*, Elsevier, 520 pp., 2005

15. Slováček D., *Performance modelling of multiprocessor parallel systems*, ICSC Leden 2012, Kunovice, Czech republic (in print).

Primary Paper Section: I

Secondary Paper Section: IN, JD, BA