IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4790

Adaptation of a Parallel Processing Technique Used to Solve a Physics Problem to a Computer Network Management Application.

Dennis Guster, Renat A. Sultanov, and Jim Q. Chen
guster@stcloudstate.edu, sultanov@underdark.bcrl.stcloudstate.edu, jchen@stcloudstate.edu
320-255-4961, 320-203-6074(fax)
St. Cloud State University, Business Computing Research Laboratory (BCRL)
720 4th Ave S., St. Cloud, MN 56301

1. INTRODUCTION

Parallelization of numeric calculations and development of new parallel algorithms is an important aspect for successful computational investigations in many fields of science. Recently the advantages of parallel processing have been recognized as applicable in the business world as well [1]. For many computational problems a single processor can easily provide timely results. However, there are certain classes of problems in which a single processor alone is not adequate. It may be that the computations or the I-O are so intensive that the single processor will need help to complete the functions quickly enough to be of value to the end-user [2].

If that is the case, parallel processing resources are more readily available and effective than they were even five years ago [3]. The combination of high speed networks such as 100BASETX, high end workstations, clustering software and middleware protocols such as MPI (message passing interface) allow a group of high end workstations connected by a LAN and configured with the appropriate software to become a loosely-coupled scalable parallel computing architecture [4]. In the past, some of the hesitation to employ parallel processing was related to its high cost. Configuring existing equipment to serve in a dual role negates much of that argument. Therefore, if an appropriate software candidate for parallel processing is identified the next concern is often related to software development. In other words, how is that software going to be effectively broken into subparts? Past work has shown three outcomes when implementing parallel processing algorithms in regard to speed: parallel is faster than serial, serial is faster than parallel, there is no significant difference between serial and parallel [5]. Obviously if the first outcome is not realized parallel processing is not worth pursuing. Developing parallel processing algorithms can be a time consuming and frustrating task and often there is no guarantee that it will speed up processing. Therefore, when trying to solve a complex problem that is a candidate for parallel processing it is often beneficial to investigate if similar algorithms exist. If so it may be efficient to try to adapt a proven algorithm rather than devise one from scratch. Therefore, this paper is designed to serve as a case study in which a proven scientific algorithm is adapted to a management problem. Specifically, the case selected is a parallel plotting algorithm used to solve a semiquantal few-body physics problem, which in turn is modified to solve a packet inter-arrival time distribution problem in the network management domain.

2. THE PHYSICS PROBLEM

It is common knowledge that parallel-methods and algorithms are highly effective for iterative and/or self-consistent mathematical algorithms, because one can bound the competitive processes in a parallel way. For example, for semiquantal systems in atomic and chemical physics one gets at least two degrees of freedom. The first degree of freedom involves quantum dynamics of lighter particles, which could be described as a "fast susbsystem" (for example electrons in molecules). The second degree of freedom is the dynamics of relatively heavier particles, which could be described as a "slow subsystem" (for example nuclei in molecules). A tool for self-consistent integration of both parts of the problem is the Pechükas method [6]. In this work we develop a parallel computer algorithm for semiquantal calculations within the Faddeev Hahn-type few-body equations [7] and the Pechukas self-consistent procedure [6] for describing the rearrangement of atomic processes 2 + (1,3) —> (2,3) + 1, with one light particle of charge Z_3 = -1 and two heavy particles 1 and 2 of charges +1. These equations are treated by means of an adequate coupled channel expansion (CCA). The basics of this method have been published in [7] and we refer readers interested in further details to this paper. The transfer cross section of the three-particle collision is described by the following equation [6]:

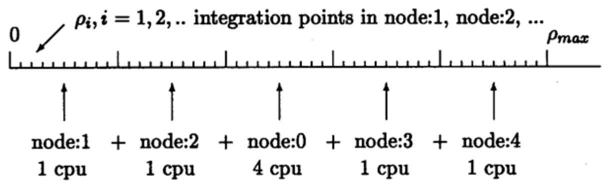
$$(1) \quad \sigma_{tr}(E) = \int_0^\infty (d\sigma_{cl}/d\Omega)\,|C_n(\rho)|^2 d\rho \approx \sum_{i=1}^{N_\rho} (d\sigma_{cl}/d\Omega)_i\,|C_n(\rho_i)|^2 w_i\ ,$$

When reduced we solve a set of coupled differential equations. When varying the impact parameter ρ_i it is found that the coefficients are independent and therefore one can solve the differential equations in completely parallel way and carry out the integration in (1) quicker.

3. METHOD OF PARALLELIZATION

Let us focus on the algorithm. Here we describe the parallel aspects of the strategy we have implemented and tested. Our program is written in C using MPI functions [8]. The code was run on the Sun Microsystems machines at the BCRL at St. Cloud State University, MN. The interprocessor topology consisted of one main machine with 4 processors and four satellites having only one processor each. We make use of the multiple processors by distributing the data according to the radial coordinate, that is the impact parameter ρ. Let $N\rho$ be a number of processors, labeled as $\kappa = 0,1, \ldots N\rho\text{-}1$. Hence, when $N\rho$ is the size of the basis in the impact parameter ρ ($\imath$), ($\imath$ = 1, $N\rho$) each processor will get $\eta\ \rho = N\ \rho\ /\ N\ p$ points along ρ. The vectors Cn (ρ ($\imath$)) are partitioned into N ρ equal sections and distributed to the various processors together with other relevant data. The initialization and set up part of the program occurs simultaneously on all processors. After that, the main calculations, a set of differential equations for a given ρ ($\imath$) is started. These computations are completely local to each processor. However, the final integration on ρ requires communication between the processors.

Fig. 1. Parallel solution of differential equations for each r on the five SUN-machines.


0
$\rho_i, i = 1, 2, ..$ integration points in node:1, node:2, ...
ρ_{max}
node:1 + node:2 + node:0 + node:3 + node:4
1 cpu 1 cpu 4 cpu 1 cpu 1 cpu

Each machine, κ, contains that portion of the quantum amplitude Cn (ρ (ί)) for the final computing (integration) of the cross sections. Fig. 1 shows the parallel processing structure. [7]

4. CONVERSION TO A COMPUTER NETWORK MANAGEMENT PROBLEM

For parallel processing to be effective a fairly high level of computational intensity is required. In searching for a problem in computer network management that meets this intensity requirement the process of forecasting packet workload offers some degree of promise. Often network managers examine "what if' scenarios with the aid of workload simulation software. The heart of that software is the calculation of simulation packet inter-arrival rates with varying degrees of intensity. In other words, what happens to response time if the mean packet inter-arrival time is reduced from .0004 seconds to .000008 seconds? Most network managers do not have the time or resources to maintain extensive packet logfile data. So often the only baseline data available to them is summary information such as mean arrival rates and how many packets arrived in a given day. This limited information can be very valuable in "what if' scenarios if the simulated arrival times are generated according to some form of exponential distribution based on the mean and number of packets received during that day. This can be done with a random number generator and when linked to simulation software this process can provide much more robust results than comparing means from day to day. Our inter-arrival rate distribution generator is built to generate data if given the mean and number of arriving packets for a given day. It is hypothesized that as the number of packets slated to arrive on any given day increases the probability that parallel processing will provide a quicker solution than serial processing increases. It is further hypothesized that if the appropriate level of intensity is reached that the parallel processing advantage realized will closely match the ratios obtained in the physics problem because it is built on the same MPI distribution logic.

5. RESULTS

The effectiveness of the parallel processing algorithm applied to the physics problem is depicted in Table 1. All temporal values are reported in standard wall clock time as returned from the MPI_WTIME function. The table is designed to illustrate how the algorithm scales when increasing the number of processor from four to eight. Both two and three iteration examples are provided. In each case approximately linear behavior was observed. Doubling the number of integration points in π caused the time to approximately increase by a factor of two.

Table 2 shows the scaling results when the distribution algorithm was applied to the computer network management problem. The distribution problem becomes a ring oriented persistent communication problem. The program generates a random observation from an exponential distribution with a mean m:

$$\chi = -m \log (1.0 — \mu), \quad (2)$$

where m = 2.0, *log* is natural logarithm and μ is the unit random number. Furthermore, the program uses MPI calls to manage the flow of numbers around the physical ring. The random number generator, which provides seed numbers for the mean calculations was programmed to generate first

Table 1. Time scaling (see) with number of processors: 2 and 3 Pechukas's classical iterations [6, 7]

number of CPU N_p	t_2 sec 2-iterations	t_3 sec 3-iterations
04	201.2	282.2
05	172.4	252.2
06	140.2	188.2
07	111.2	163.2
08	88.2	122.2

2,000 then 20,000 numbers so that different levels of intensity could be observed. Again linear dependence was observed in all columns reported. However, the magnitude was the inverse of what was desired. In other word, as processors were added the total execution time increased. This was true for both the 2,000 and 20,000 observation levels. Curiously, the times for the 20,000 observation trials were about 10 times greater than the 2,000 observation trials. This held true for both the five and ten logical channel communication models. The fact that opening five more communication channels about doubled the execution time further supports the observed linear patterns.

It is clear that both of the hypotheses must be rejected, increasing the number of observations did not make the algorithm any more efficient and in no way did the adapted algorithm come close to the excellent results achieved in the physics problem. This could be attributed to the more complex differential equations employed in the physics problem on the application level. The processor distribution portion of the algorithm was successfully adapted, but the application chosen (generating packet distributions) lacked the complexity needed to be an appropriate candidate for parallel processing. Therefore, it would seem appropriate to search out more complex business applications and test their suitability for the processor distribution algorithms employed herein.

6. IMPLICATIONS FOR MANAGERS

Although this study was not successful in adapting a scientific algorithm to solve a computer network management problem it did generate a useful frame work from which to pursue additional research. The distribution portion of the algorithm can be easily adapted to other business application problems. Perhaps if a problem of adequate intensity is identified then more promising results may be realized. It also might be interesting to try the same problem using other parallel processing mechanisms. Perhaps PVM (parallel virtual machine), which uses a more efficient communication protocol for processor-to- processor communication than MPI would offer better results. This study certainly further proved that not all problems are appropriate for parallel processing.

Table 2.Time scaling (sec) with number of processors N p and number of rings Nr for parallel generation of 2,000 and 20,000 random observations from an exponential distribution.

Number of CPU's	Number of Rings			
	$N_r = 5$		$N_r = 10$	
N_p	2,000	20,000	2,000	20,000
04	0.038	0.374	0.076	0.750
05	0.093	0.822	0.168	1.582
06	0.139	1.260	0.258	2.381
07	0.188	1.709	0.348	3.201
08	0.237	2.152	0.437	4.017

REFERENCES

[1] Paprzycki, M. and Kalczynski, P. "Supercomputers on their way to business", Proceedings of the Information Systems Architecture and Technology Conference, Szklarska, Poland, Sept 20-21, 2001.

[2] Pfister, G. In Search of Clusters, Prentice Hall, 2nd ed. 1998.

[3] Buyya, R. High Performance Cluster Computing, Vol. 1, Prentice Hall, 1999.

[4] Hwang, K. and Xu, Z. Scalable Parallel Computing: Technology, Architecture and Programming, McGraw-Hill, 1998.

[5] Guster, D. and Madison, D. "Integrating Parallel Processing into the Microcomputer Education Curriculum". A paper presented at the Midwest Decision Sciences Institute, Indianapolis, IN, May 1-3, 1991.

[6] Pechukas P., Phys. Rev. 181, 166 (1969); 181, 174 (1969).

[7] Sultanov R. A., Sandhas W., Belyaev V. B., Eur. Phys. J. D 5 33 (1999); Sultanov R. A., Phys. Rev. A 50, 2376 (1994).

[8] Gropp W., Lusk E., Skjellum A., Using MPI: Portable Parallel Programming with the Message- Passing Interface, MIT Press, 1999.