Available online at www.sciencedirect.com
SCIENCE DIRECT
Nuclear Instruments and Methods in Physics Research A 559 (2006) 306–309
NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH Section A
www.elsevier.com/locate/nima

Recent developments in parallelization of the multidimensional integration package DICE

F. Yuasa[a,*], K. Tobimatsu[b], S. Kawabata[a]

[a]*High Energy Accelerator Research Organization, KEK, 1-1 OHO Tsukuba, Ibaraki 305-0801, Japan*
[b]*Kogakuin Univ., 1-24-2 Nishi-Shinjuku, Shinjuku-ku, Tokyo 163-8677, Japan*

Available online 22 December 2005

Abstract

DICE is a general purpose multidimensional numerical integration package. There can be two ways in the parallelization of DICE, "distributing random numbers into workers" and "distributing hypercubes into workers". Furthermore, there can be the combination of both ways. So far, we had developed the parallelization code using the former way and reported it in ACAT2002 in Moscow. Here, we will present the recent developments of parallelized DICE in the latter way as the 2nd stage of our parallelization activities.

PACS: 02.60.Jh

Keywords: Parallel computing; Multidimensional numerical integration; Monte Carlo integration; DICE; MPI

1. Introduction

Recently it is not rare to calculate the cross-sections of the physics processes with over 6 final state particles in the tree level. In such calculations, there may appear singularities close to diagonal integral region and sometimes it is very difficult to find a good set of variable transformations to get rid of the singularities. For the one-loop and beyond the one-loop physics processes, when we try to carry out the loop calculation only in the numerical approach, we need several multidimensional integration packages or another integration method to compare the numerical results to check them. For such a request, DICE has been developed by Tobimatsu and Kawabata. It is a general purpose multidimensional numerical integration package.

1.1. The non-parallelized version of DICE

The first version of DICE [1] appeared in 1992 and is a scalar program code. In DICE, the integral region is divided into $2^{N_{\mathrm{dim}}}$ hypercubes repeatedly according to the division condition. To evaluate the integral and its variance in each hypercube, DICE tries two kinds of sampling methods, a regular sampling and a random sampling as

1. Apply regular sampling and evaluate the contribution.
   And then check the division condition is satisfied or not.
2. Apply 1st random sampling and evaluate the contribution.
   And then check the division condition is satisfied or not.
3. Apply 2nd random sampling and evaluate the contribution.

For an integrand with singularities the number of above repetitions becomes huge so rapidly and the calculation time becomes a long time. To reduce the calculation time, the vectorized version of DICE (DICE 1.3 Vh [2]) has been developed in 1998 for vector machines. In the vector program code, the concept of workers and the queuing mechanism are introduced. This vectorized DICE has succeeded in reducing the calculation time for the integration even when the integrand has strong singularities.

Today, however, the vector processor architecture machines have dropped off and instead the parallel processor architecture machines have become common in the field of high energy physics. Moreover, the

*Corresponding author. Tel.: +81 29 8645479.
E-mail address: fukuko.yuasa@kek.jp (F. Yuasa).

0168-9002/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.nima.2005.11.218

cost-effective PC clusters running Linux with a distributed memory or a shared memory are widely spread. Thanks to this rapid rise of PC clusters with the parallel library such as MPI [3] or with OpenMP [4], the parallel programming is very familiar to us.

2. Parallelization

2.1. Profile of DICE

To get a good efficiency in the parallelization, it is important to know which routines are time-consuming. UNIX command `gprof` is a useful tool to know it. Table 1 shows an example output of `gprof` command for the calculation of the integration by the non-parallelized DICE. This calculation is done on the Alpha 21264 processor (700 MHz clock speed) machine running Linux and the compiler used is Compaq Fortran. In Table 1, the most time-consuming routine is `elwks` and is called in `func`. In `elwk` and `func` the integrand function is given. The subroutine `func` is called repeatedly in `regular`, `random1` and `random2` to evaluate the integrand. Here, `vbrndm` is a routine to generate random numbers and is called in both `random1` and `random2`.

In summary, it is expected that distributing the calculations in `random1` and `random2` into workers (processors) may be efficient to reduce the calculation time.

2.2. Algorithm

For the integrand with strong singularities, the region is divided into a large number of hypercubes and a large number of random numbers are required to get the integral results with the requested errors. Therefore, there can be two ways in the parallelization, the way of distributing random numbers and the way of distributing hypercubes to workers.

As the 1st step we have started the parallelization of DICE with the former way, distributing random numbers. The schematic view of the algorithm with the former way is shown in Fig. 1. There, it is shown how random numbers are distributed into workers in `random1` and `random2`. The merit of this approach is not only that the algorithm is very simple as shown in Fig. 1 but also that the overhead due to the data transfer or the load unbalancing among workers is small. The efficiency of this parallelization has showed very good performance. The result of the efficiency measurement by this parallelization way was presented at ACAT2002 at Moscow [5].

As the 2nd step, here in this paper, we present the parallelization with the latter way, distributing hypercubes into workers. As the 3rd step, the final step, we have a plan of the combination of both ways.

3. Implementation

In this parallelization, hypercubes are distributed into workers. After the evaluation, the results are gathered to the root process (for example, worker 1). And then the root process scattered the results to all workers. In Fig. 2, a schematic view of how calculations are distributed into workers is shown.

In our implementation we use Fortran compiler since DICE is written in `Fortran` and we chose `MPI` [3] as the parallel library.

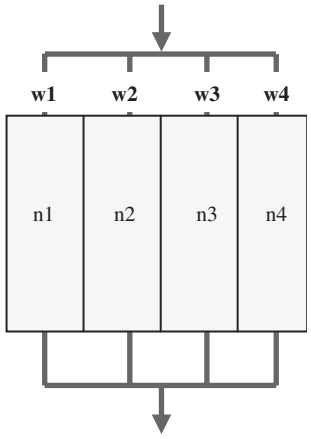

Fig. 1. Schematic view of the algorithm of parallelized DICE with distributing the random numbers into workers, w1, w2, w3 and w4, for example. Each worker is responsible for the part of random numbers in the routines, `random1` and `random2`. The total number of random numbers is the sum of random numbers treated in each worker as $N_{\text{total}} = n1 + n2 + n3 + n4$.

Table 1
`gprof` output: flat profile of non-parallelized DICE

Time (%)	Cumulative time (s)	Self time (s)	Calls	Self (μ/call)	Total	Name of routines
82.95	7.60	7.60	26214	0.29	0.29	`elwks_`
12.41	8.73	1.14	26214	0.04	0.33	`func_`
2.52	8.96	0.23	3072	0.08	0.08	`vbrndm_`
0.93	9.05	0.08	1536	0.06	2.80	`random2_`
0.92	9.13	0.08	1536	0.05	2.79	`random1_`
0.13	9.14	0.01	1638	0.01	0.34	`regular_`

This calculation of the integration is done on the Alpha 21264/700 MHz machine by the Compaq Fortran for Linux. Total CPU time required was 9.16 s in total. This integration was done with expected error = 10%.