Redbooks Paper

Tzy-Hwa Kathy Tzeng
Yinhe Cheng
Mitsunori Ogihara
Andrew I. Brooks

Performance Tuning of ICED on POWER4 Platforms

A striking example of the effectiveness of High Performance Computing was recently demonstrated on an IBM® POWER4™ platform. The project involved a biological application called “Independently Consistent Expression Discriminator (ICED).” ICED is a sample classification application for microarray data analysis that helps identify disease-relevant genes for medical research. After enabling and tuning ICED on an IBM @server pSeries® 690, a job that used to take more than three days on Microsoft® Windows® now takes about 10 minutes on AIX®.

The authors of this Redpaper

This Redpaper was produced by a team of specialists from around the world.

Tzy-Hwa Kathy Tzeng is a Life Sciences Software Engineering Consultant at IBM High Performance Computing Solutions Development. She received her Ph.D. from Iowa State University and has several years of experience in the area of High Performance Computing.

Yinhe Cheng is a Ph.D. candidate in the Department of Computer Sciences, University of Rochester. She is one of the primary designers and developers of ICED. Her research interests include bioinformatics, machine learning, and data mining.

Mitsunori Ogihara received a Ph.D. in Information Sciences from Tokyo Institute of Technology, Japan, in 1993. Since 1994, he has been on the faculty at the University of Rochester. He is currently a Professor of Computer Science and of the Center for Aging and Developmental Biology. He is also chair of the Department of Computer Science. His current research interests include computational complexity theory, data mining, machine learning, molecular computing, bioinformatics, and music information retrieval.

Andrew I. Brooks is Director of the Functional Genomics Center at the University of Rochester Medical Center. He received his doctorate from the University of Rochester and currently holds an appointment in Environmental Medicine as an Assistant Professor. Dr. Brooks' main areas of interest include deciphering the molecular mechanisms that underlie memory and learning, developing novel informatics approaches to better understand complex genomics-based data sets, and technology development in the areas of transcript and protein profiling.

Abstract

Independently Consistent Expression Discriminator (ICED) is a sample classification application for Microarray data analysis. While it is based on a robust algorithm, it can take a long time, sometimes even days, to complete a cross-validation test for large data sets on its development platform, Windows. A faster processing time is highly desired.

ICED was enabled and optimized on an IBM POWER4 system. The benchmark results indicate that the optimized ICED code runs 2.6 to 4.0 times faster than the original code on Windows. The same optimized code on AIX with a single processor achieved a 3.5 to 4.1 times improvement over Windows with the same processor speed.

ICED was then parallelized using OpenMP. The parallelized code has good scalability on AIX, especially for large jobs. A typical cross-validation of a large data set achieved a 15.8 times speedup with 16 CPUs and a 31.3 times speedup with 32 CPUs on a p690+ system. It used to take more than three days to perform a cross-validation test on a large data set using the original code on Windows. Now, it only takes approximately 10 minutes for the same data set using the optimized code and 32 CPUs on AIX. Our results also suggest that the performance improvement and scalability increases when the size of the data set increases and the percentage of genes considered is higher.

Introduction

Independently Consistent Expression Discriminator (ICED) [1] is one of the most robust applications for analyzing high-throughput gene expression data. This application is used to build binary predictors for clinical diagnosis based on genes that can differentiate disease states and to identify disease-relevant genes for medical research.

ICED has five components:

i. Normalization of raw data

ii. Assignment of weights to genes for two different classes

iii. Calculation of votes from determined number of genes; generation of prediction strength

iv. Determination of the optimal numbers of predictor genes of each class for class distinction

v. Calculation of prediction strength for unknown samples

ICED has the following three functions:

- Training
- Testing
- Cross-validation

In training, a predicting system is built based on a labeled data set. Training consists of components (i) to (iv). In Testing, unknown samples are predicted by a pretrained system. Testing consists of component (v). Cross-validation tests the accuracy and robustness of the system on a specific data set [4]. It repeats training on random selected subsets of the data set and testing on their complementary set. In this study, we used leave-one-out cross-validation [4] in all the cross-validation benchmarks.

Complexity analysis of the algorithm shows that:

Training: $O(p^2 \times g^3 \times s)$
Testing: $O(g \times s)$
Cross-validation: $O(p^2 \times g^3 \times s^2)$

Where:

- *s* is the number of samples in the data set.
- *g* is the number of genes in the data set.
- *p* is the portion of the total number of genes that the program considers in the determination of optimal discriminator numbers; *0<p<1*.

Due to the complexity of the algorithm, both training and cross-validation can be very time consuming, especially cross-validation. When the number of genes or the number of samples increases, statistically, the predicting accuracy would increase; on the other hand, the computation time increases exponentially. The run time on the development platform, Windows, sometimes can be several days. Running ICED on a platform with higher computation capacity and good scalability is highly desirable.

The purpose of this study is to enable, optimize, and parallelize ICED on POWER4 systems so that the computation time can be reduced. Benchmarks were performed to illustrate the performance improvement of optimized code and good scalability of parallelized ICED code on POWER4 platforms.

Methods and systems

Two data sets are used in the benchmarks:

- The Leukemia data sets were generated at the Whitehead Institute and the Center for Genome Research at the Massachusetts Institute of Technology [2, 3]. This data set has a total of 72 samples: 25 acute myeloid leukemia (AML) samples and 47 acute lymphoblastic leukemia (ALL) samples. For each sample, the expression levels of 7129 genes are measured. In our experiment, we use ICED to distinguish the samples of AML and the samples that of ALL.
- The St. Jude data set was published by Yeoh et al. [5]. The data set consists of 248 samples that belong to six subclasses of Leukemia: T-ALL (43 samples), E2A-PBX1 (27 samples), BCR-ABL (15 samples), TEL-AML1 (79 samples), MLL (20 samples), and Hyperdiploid (64 samples). For each sample, the expression levels of 12625 genes are measured. In our experiment, we use ICED to distinguish samples of TEL-AML and the samples that are not TEL-AML.

The systems used for our performance comparison are IBM IntelliStation® MPro, p630+, p690, and p690+. The detailed configurations are listed in Table 1.

Table 1 System and hardware configuration

	IntelliStation MPro	**p690**	**p630+**	**p690+**
Processor	1.7 GHz Pentium® 4	1.3 GHz POWER4	1.45 GHz POWER4+™	1.7 GHz POWER4+
Number of processors	1	32	4	32
Memory	1 GB	160 GB	16 GB	64 GB
OS	Windows 2000	AIX 5L™ Version 5.1	AIX 5L V5.1	AIX 5L V5.1
C/C++ compiler	Microsoft Visual C++ 6.0	VisualAge® C++ V6.0	VisualAge C++ V6.0	VisualAge C++ V6.0

ICED was developed using Microsoft Visual C++ on Windows 2000. It was ported to AIX using the IBM VisualAge C++ compiler and tuned using compiler optimization on POWER4. Xprofiler was used to identify the most time-consuming codes. The performance was further improved by optimizing the memory management, reducing I/O for cross-validation, converting recursive functions into non-recursive functions, and bringing frequently used small functions inline. The optimized ICED code was then parallelized using OpenMP on AIX. The optimized code was also tested on Windows for the purpose of comparison.

To efficiently parallelize ICED, the structure of the program was modified. Example 1 illustrates the original structure and Example 2 on page 5 illustrates the program structure after parallelization. Block C consumes most CPU time and will benefit most from parallelization. The calculation of the average of votes is part of Block C but cannot be parallelized easily.

Example 1 Structure of the training program of ICED before parallelization

```
Start Program
BlockA Read database of gene expression data

BlockB For each gene
               Generate weight for the gene
       End for
       For i=0;i<p*gene_num;i++
               For each sample in the dataset
                        Generate vote1(i) of the ith gene in class1 list for the sample
               End for
       End for
       For j=0;j<p*gene_num;j++
               For each sample in the dataset
                        Generate vote2(j) of the jth gene in class2 list for the sample
               End for
       End for

BlockC  For i=0;i<p*gene_num;i++
                For j=0;j<p*gene_num;j++
                        For each sample k in the dataset
                                Summarize the votes of i genes for class1 and j genes for class2, and
generate the prediction strength for the sample.
                                PS(i,j,k)=(ave_vote1(i,k)+ave_vote2(j,k))/
                                 (ave_vote1(i)+ave_vote2(j))
                                ave_vote2(j,k)=(ave_vote2(j-1,k)*(j-1)+vote(j,k))/j
                                ave_vote1(i,k)=(ave_vote1(i-1,k)*(i-1)+vote(i,k))/i
                         End for
                         Calculate fitness score F(I,j) for pair (I,j)
                         If F(i,j) is better than the best fitness score so far
                                  Update (best1, best2)=(i,j)
                         End if
                End for
        End for

BlockD  For i=0;i<best1;i++
                Output weight of the i'th gene in class1 list
        End for
        For j=0;j<best2;j++
                Output weight of the j'th gene in class2 list
        End for
End Program
```

To parallelize the code, the calculation of the average of the votes was divided into two steps: calculation of the sum of the votes and calculation of the average. The calculation of the sum of the votes cannot be easily parallelized because the calculation depends on previous results. However, it consumes relatively insignificant time, and thus no further attempt was made for parallelization. In the parallelized code, the calculation of the sum of the votes is executed before the triply nested for-loop of Block C. Therefore, the for-loop can be efficiently parallelized. Block A and Block D are I/O related, thus no parallelization was done for these two parts.

Example 2 Structure of the training program of ICED after parallelization

```
Start Program
BlockA Read database of gene expression data

BlockB For each gene
               Generate weight for the gene
       End for
       For i=0;i<p*gene_num;i++
               For each sample in the dataset
                        Generate vote1(i) of the ith gene in class1 list for the sample
               End for
       End for
       For j=0;j<p*gene_num;j++
               For each sample in the dataset
                        Generate vote2(j) of the jth gene in class2 list for the sample
               End for
       End for

BlockC  For i=0;i<p*gene_num;i++
                For j=0;j<p*gene_num;j++
                         For each sample k in the dataset
                                  Summarize the votes of i genes for class1 and j genes for class2, and
generate the prediction strength for the sample.
                                  PS(i,j,k)=(ave_vote1(i,k)+ave_vote2(j,k))/
                                   (ave_vote1(i)+ave_vote2(j))
                                  ave_vote2(j,k)=(ave_vote2(j-1,k)*(j-1)+vote(j,k))/j
                                  ave_vote1(i,k)=(ave_vote1(i-1,k)*(i-1)+vote(i,k))/i
                          End for
                          Calculate fitness score F(I,j) for pair (I,j)
                          If F(i,j) is better than the best fitness score so far
                                    Update (best1, best2)=(i,j)
                          End if
                End for
        End for

BlockD  For i=0;i<best1;i++
               Output weight of the i'th gene in class1 list
        End for
        For j=0;j<best2;j++
                Output weight of the j'th gene in class2 list
        End for
End Program
```

Results

The optimized ICED was benchmarked on AIX and Windows. The performance improvement (PI) is defined as the ratio of the elapsed time of the longer run (t_l) over the elapsed time of the shorter run (t_s):

$$PI = t_l / t_s$$

Two different data sets, St. Jude [5] and Golub [2, 3], were used. The two most time-consuming functionality, training, and cross-validation [1, 4] were evaluated. Jobs of different sizes are generated by selecting different *p* values, which is the portion of the total number of genes that the program considers in the determination of optimal numbers of discriminator genes. The higher the *p* value, the more accurate the prediction, and the longer the execution time.

Performance comparison of optimized and original code on Windows

Although the code was optimized on AIX, the optimized code achieved a 2.6 to 4.0 times improvement compared to the original code on Windows. It suggests that the larger the jobs, the more the improvement. Figure 1, Figure 2, and Figure 3 illustrate the performance comparison of the optimized code versus the original code.

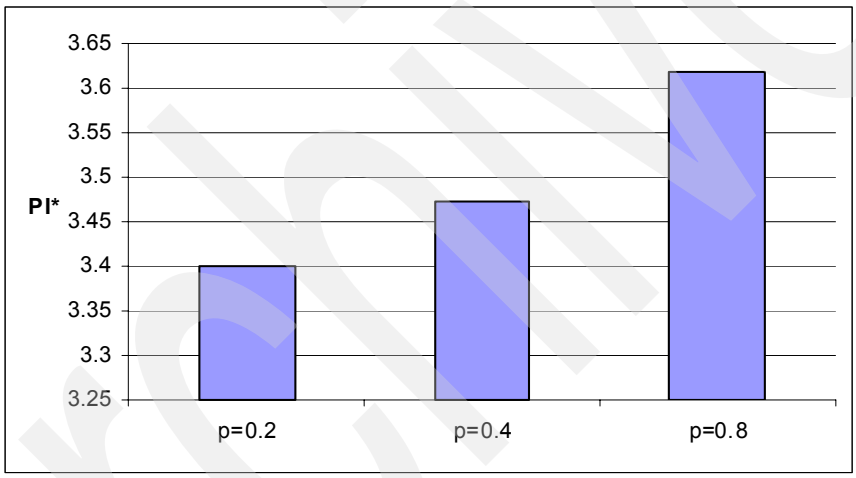

Figure 1 Optimized versus original code on Windows: Training of St. Jude data set

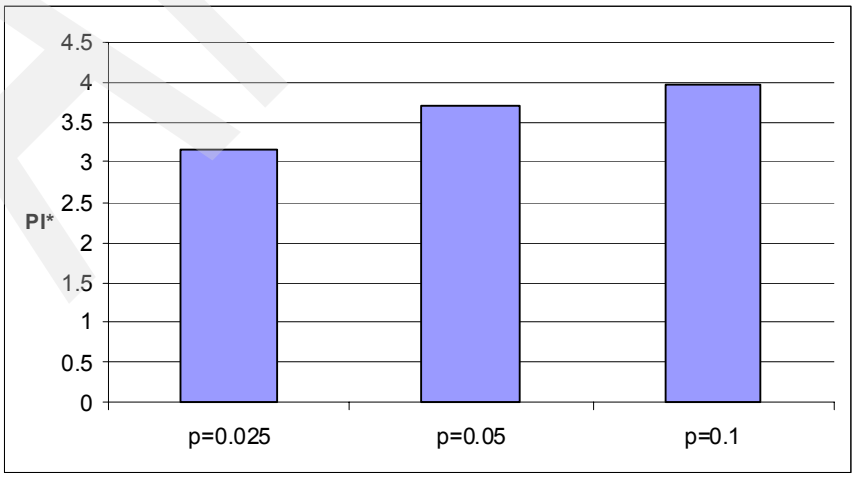

Figure 2 Optimized versus original code on Windows: Cross-validation of St. Jude data set

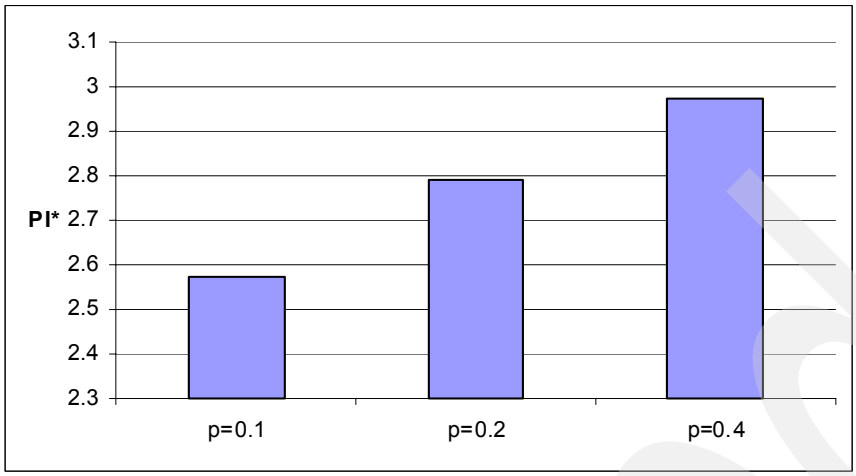

Figure 3 Optimized versus original code on Windows: Cross-validation of Golub data set

*** Note:** PI equals the run time of the original code on Windows/run time of the optimized code on Windows.

Performance comparison of optimized code on Windows and AIX

Figures 4 through 6 show the performance of optimized ICED on AIX compared to Windows. All tested AIX systems display higher performance than Windows, with AIX outperforming Windows by factors ranging from 2.5 to 4.1. The p690+ we used has the same CPU clock rate as the IntelliStation MPro (see Table 1 on page 3). The performance improvement of p690+ over IntelliStation MPro is in the range of 3.5 to 4.1 with a single processor.

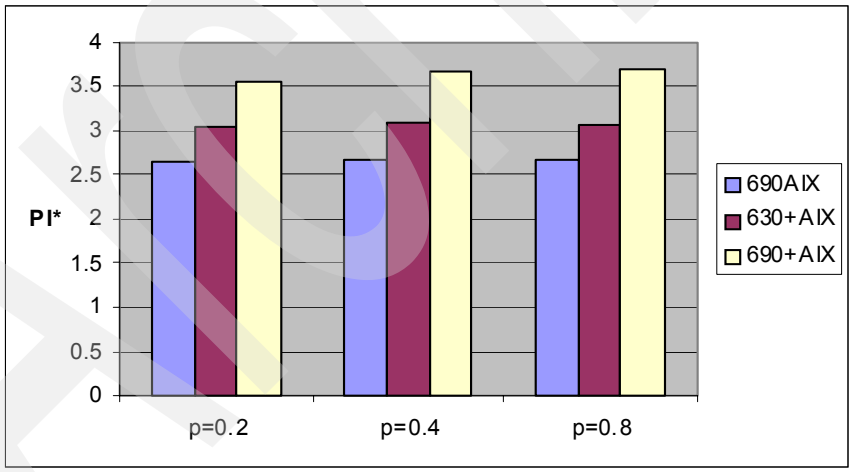

Figure 4 Optimized code on Windows versus AIX: Training of St. Jude data set

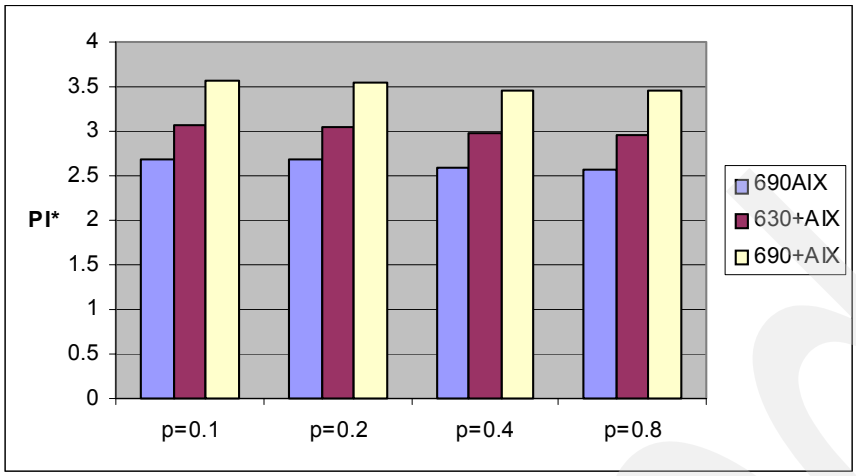

Figure 5 Optimized code on Windows versus AIX: Cross-validation of Golub data set

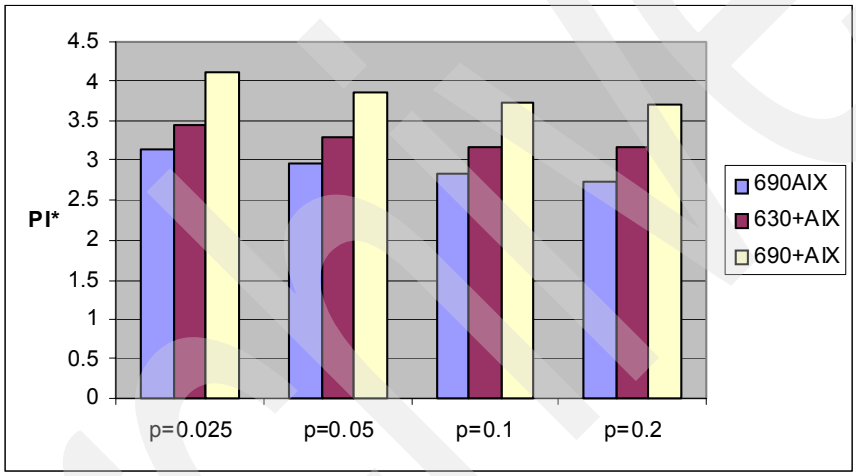

Figure 6 Optimized code on Windows versus AIX: Cross-validation of St. Jude data set

*** Note:** PI equals the elapsed time of the optimized code on Windows/elapsed time of the optimized serial code on AIX.

Scalability of parallelized ICED on AIX

The parallelized code has good scalability on AIX, especially for large jobs. To determine the scalability, Speedup (S) was defined as the ratio of the elapsed time from serial run (t_s) over the elapsed time from parallel run (t_p):

$$S = t_s / t_p$$

The elapsed times for ICED using different numbers of CPUs on p690, p690+, and p630+ are shown in Tables 2 through 4. These compare the original and optimized runs on IntelliStation and POWER4 platform.

Table 2 Elapsed time of ICED: Training of St. Jude data set

	Original code	Optimized code	1 CPU	2 CPUs	4 CPUs	8 CPUs	16 CPUs	32 CPUs
	IntelliStation		*p690*					
p=0.2	15m28.14s	4m33s	1m43.23s	56.84s	32.52s	20.18s	14.07s	11.03s
p=0.4	1h1m5.66s	17m35.57s	6m34.66s	3m24.63	1m46.57s	57.41s	33.14s	20.99s
p=0.8	4h12m28.55s	1h9m44.57s	26m5.15s	13m12.73s	6m42.48s	3m29.95s	1m52.1s	1m1.57s
	IntelliStation		*p630+*					
p=0.2	15m28.14	4m33s	1m29.24s	48.06s	27.47s			
p=0.4	1h1m5.66s	17m35.57s	5m40.9s	2m53.46s	1m29.99s			
p=0.8	4h12m28.55s	1h9m44.57s	22m42.03s	11m11.76s	5m44.41s			
	IntelliStation		*p690+*					
p=0.2	15m28.14s	4m33s	1m16.65s	41.28s	23.58s	14.64s	10.23s	8s
p=0.4	1h1m5.66s	17m35.57s	4m47.89s	2m27.89s	1m16.94s	41.76s	24.2s	15.01s
p=0.8	4h12m28.55s	1h9m44.57s	18m51.24s	9m34.55s	4m53.82s	2m31.29s	1m19.4s	42.78s

Table 3 Elapsed time of ICED: Cross-validation of Golub data set

	Original code	Optimized code	1 CPU	2 CPUs	4 CPUs	8 CPUs	16 CPUs	32 CPUs
	IntelliStation		*p690*					
p=0.1	20m4.04s	7m48.08s	2m53.54s	1m28.62s	46.04s	24.54s	13.8s	8.61s
p=0.2	1h23m18.65s	29m49.69s	11m10.59s	5m38.84s	2m52.22s	1m28.05s	45.96s	25.04s
p=0.4	5h49m37.5s	1h57m31.88s	45m7.17s	22m43.08s	11m27.3s	5m43.05s	2m54.27s	1m29.82s
p=0.8	22h55m33.52s	7h47m43.02s	3h1m6.54s	1h31m11.8s	45m51.03s	22m44.83s	11m30.24s	5m49.26s
	IntelliStation		*p630+*					
p=0.1	20m4.04s	7m48.08s	2m33.41s	1m18.26s	41.35s			
p=0.2	1h23m18.65s	29m49.69s	9m49.48s	4m58.23s	2m31.74s			
p=0.4	5h49m37.5s	1h57m31.88s	39m24.21s	19m43.57s	9m51.9s			
p=0.8	22h55m33.52s	7h47m43.02s	2h38m6.77s	1h18m40.9s	39m33.06s			
	IntelliStation		*p690+*					
p=0.1	20m4.04s	7m48.08s	2m10.95s	1m6.85s	34.77s	18.6s	10.41s	6.57s
p=0.2	1h23m18.65s	29m49.69s	8m24.93s	4m15.61s	2m9.83s	1m6.84s	34.7s	18.87s
p=0.4	5h49m37.5s	1h57m31.88s	33m55.09s	17m1.09s	8m35.83s	4m19.32s	2m11.32s	1m7.27s
p=0.8	22h55m33.52s	7h47m43.02s	2h15m15.25s	1h7m58.62s	34m22.5s	17m0.55s	8m34.01s	4m18.99s

Table 4 Elapsed time of ICED: Cross-validation of St. Jude data set

	Original code	Optimized code	1 CPU	2 CPUs	4 CPUs	8 CPUs	16 CPUs	32 CPUs
	IntelliStation		*p690*					
p=0.025	1h12m28.04s	22m49.26s	7m16.73s	3m46.73s	2m1.69s	1m9.13s	42.11s	28.29s
p=0.05	4h31m22.58s	1h12m55.26s	24m41s	12m35.74s	6m30.44s	3m24.15s	1m53.24s	1m6.14s
p=0.1	18h12m1.91s	4h34m13.13s	1h36m49.81s	49m49.44s	25m24.21s	12m50.35s	6m36.29s	3m35.08s
p=0.2	3d32m21.8s	18h1m14.86s	6h33m5.7s	3h22m12.83	1h42m6.79s	51m19.79s	25m55.88s	13m31.5s
	IntelliStation		*p630+*					
p=0.025	1h12m28.04s	22m49.26s	6m39.17s	3m27.18s	1m52.6s			
p=0.05	4h31m22.58s	1h12m55.26s	22m10.82s	11m19.11s	5m53.38s			
p=0.1	18h12m1.91s	4h34m13.13s	1h26m13.4s	44m13.25s	22m30.93s			
p=0.2	3d32m21.8s	18h1m14.86s	5h41m13.82s	2h51m30.77s	1h27m9.69s			
	IntelliStation		*p690+*					
p=0.025	1h12m28.04s	22m49.26s	5m33.69s	2m52.72s	1m32.38s	52.19s	31.46s	20.87s
p=0.05	4h31m22.58s	1h12m55.26s	18m51.82s	9m37.13s	4m58.2s	2m35.44s	1m25.8s	49.67s
p=0.1	18h12m1.91s	4h34m13.13s	1h13m40.42s	37m49.92s	19m9.78s	9m44.69s	5m0.28s	2m42.72
p=0.2	3d32m21.8s	18h1m14.86s	4h52m5.86s	2h26m59.83s	1h14m14.11s	37m25.65s	19m12.73s	10m7.2s

Figures 7 through 9 illustrate the scalability of ICED with different sizes of jobs on p690, p630+, and p690+. In all the tests, the higher the p values, the longer the execution time, and the better the scalability. It also shows that the cross-validation, which is more computation intensive than training, demonstrates better scalability. The results suggest that the larger the job, the better the scalability. As shown in Figure 8, the cross-validation of the Golub data set achieved a 15.8 times speedup with 16 CPUs and a 31.3 times speedup with 32 CPUs.

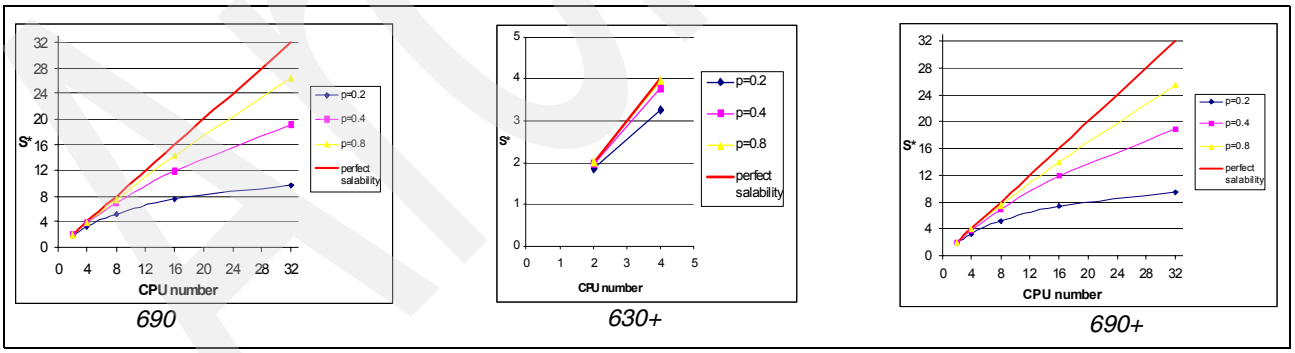

Figure 7 Scalability of ICED: Training of St. Jude data set

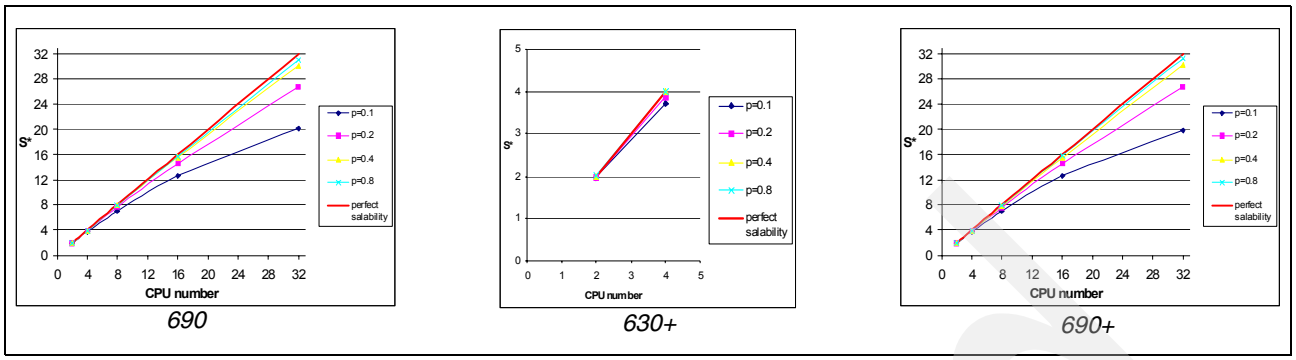

Figure 8 Scalability of ICED: Cross-validation of Golub data set

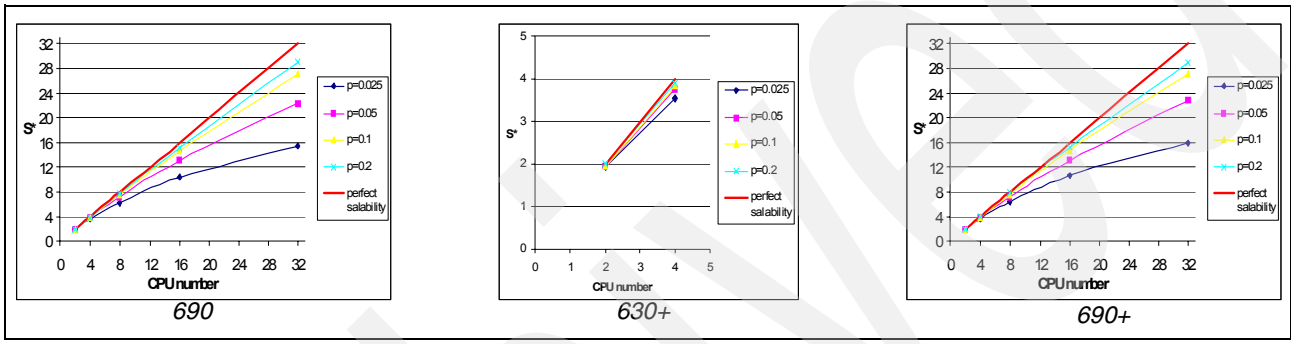

Figure 9 Scalability of ICED: Cross-validation of St. Jude data set

*** Note:** S equals the serial run time of the optimized code on AIX/parallel run time of the optimized code on AIX.

Overall performance improvement

An overview of the performance improvement is presented in this section.

Performance improvement of optimized code on AIX versus Windows

Figures 10 through 12 show the performance improvement achieved by the optimized code on p690, p630+, and p690+ systems compared to the optimized code on IntelliStation MPro. In general, larger jobs achieved better performance improvements. More than 100 times performance improvements were achieved with 32 CPUs on p690+ for the cross-validation test with higher p values, as shown in Figures 11 and 12.

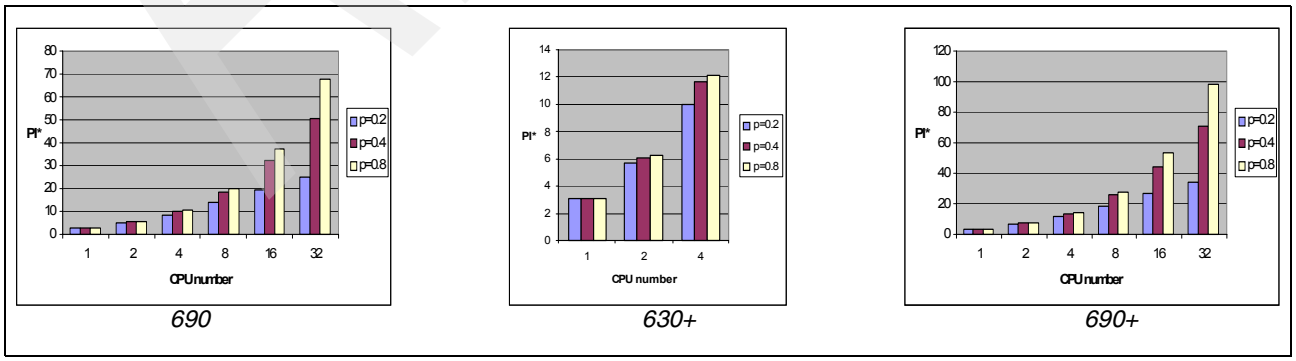

Figure 10 Optimized code on AIX versus Windows: Training of St. Jude data set

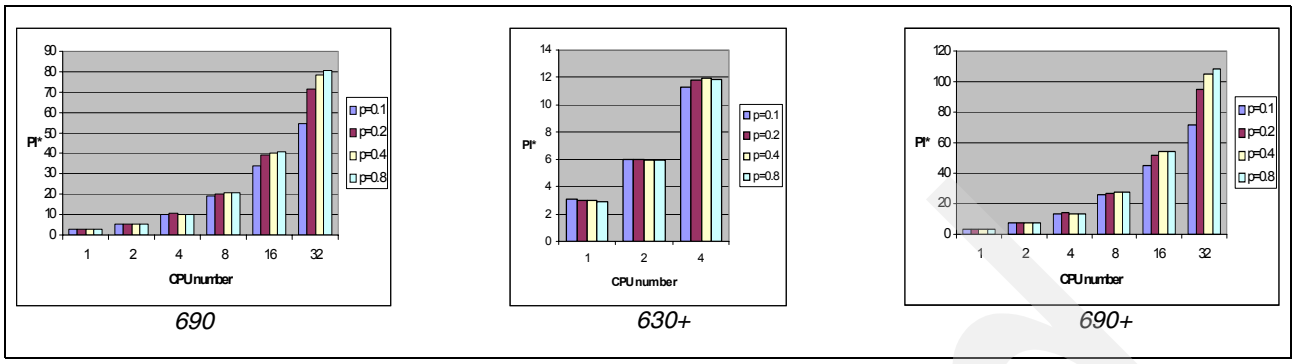

Figure 11 Optimized code on AIX versus Windows: Cross-validation of Golub data set

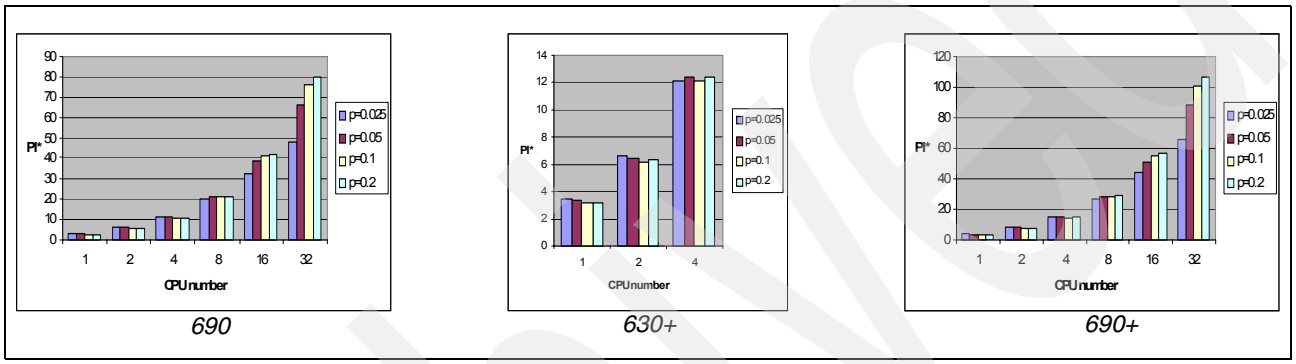

Figure 12 Optimized code on AIX versus Windows: Cross-validation of St. Jude data set

*** Note:** PI equals the run time of the optimized code on Windows/run time of the optimized code on AIX.

Performance improvement of optimized code on AIX versus original code on Windows

Figures 13 through 15 show the performance improvement achieved by the optimized code on p690, p630+, and p690+ systems compared to the original code on IntelliStation MPro. A performance improvement of 300 times 400 times was achieved with 32 CPUs on p690+ for typical test cases. It took more than three days for the original ICED to finish the cross-validation test of the St. Jude data set with *p=0.8* on the IntelliStation MPro. With 32 CPUs on p690+, this test was finished in about 10 minutes.

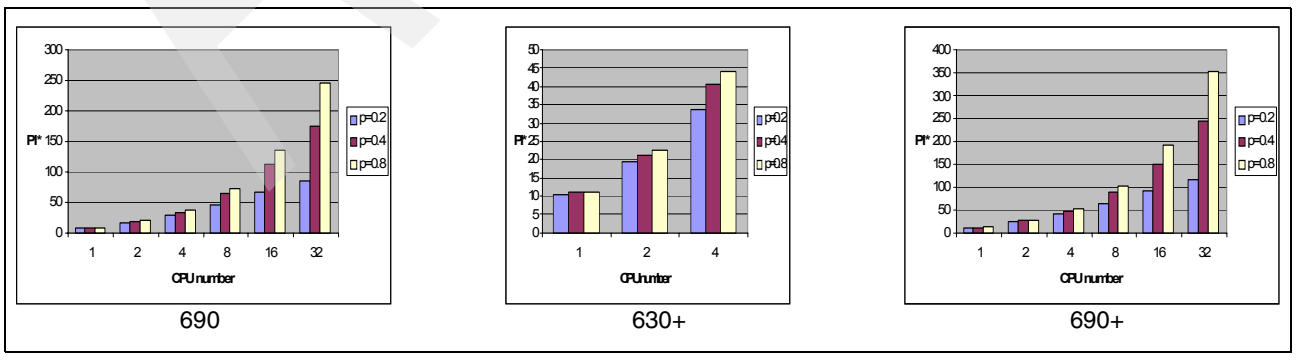

Figure 13 Optimized code on AIX versus original code on Windows: Training of St. Jude data set

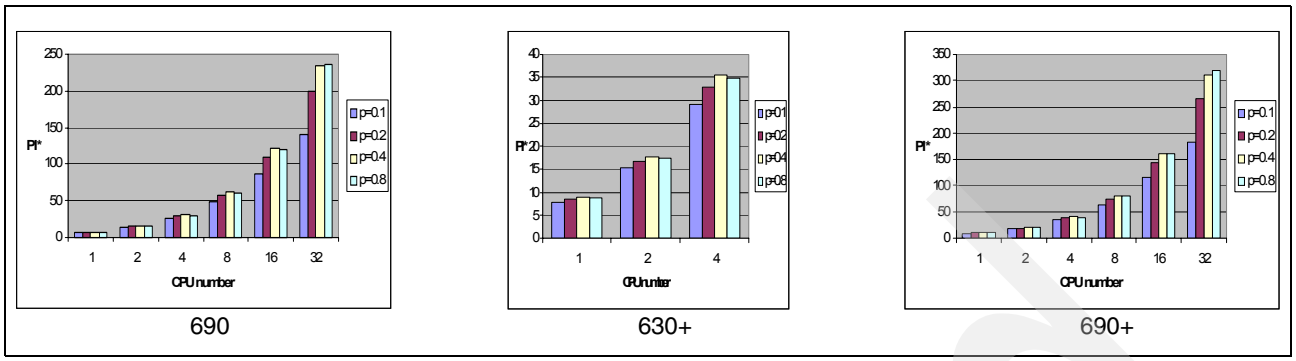

Figure 14 Optimized code on AIX versus original code on Windows: Cross-validation of Golub data set

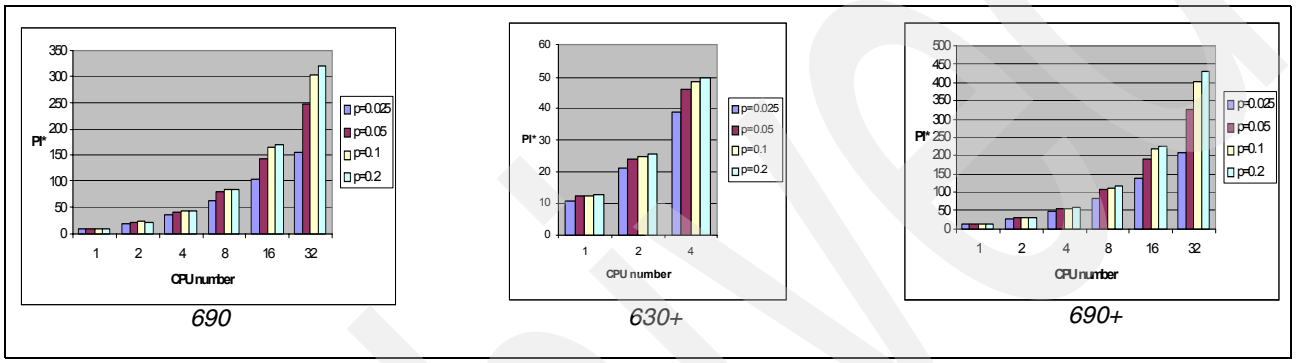

Figure 15 Optimized code on AIX versus original code on Windows: Cross-validation of St. Jude data set

*** Note:** PI equals the runtime of the original code on Windows/run time of the optimized code on AIX.

Conclusions

After porting, optimizing, and parallelizing ICED on a POWER4 system, we reached the goal of significantly reducing the computation time of the gene expression data analysis algorithm ICED. We also demonstrated the advantage of the POWER4/POWER4+ systems and its good scalability on scientific applications such as ICED. The speedup of single CPU run for ICED from 1.7 GHz IntelliStation MPro to 1.7 GHz p690+ is 3.5 to 4.1 fold with the same optimized code.

The scalability of optimized ICED on AIX is close to linear when the job is large. The longest test case we used took more than three days on IntelliStation before code optimization, and it only took 14 minutes on p690 using 32 CPUs, and 10 minutes on p690+ using 32 CPUs.

Our results suggest that when the size of the job increases, both the performance improvement of ICED from IntelliStation MPro to POWER4 systems and the scalability of ICED on POWER4 systems increase.

Acknowledgements

We would like to thank Farid Parpia for suggestions and discussion about tuning and parallelization. We would also like to thank Joyce Mak, Tina Tarquinio, Julie Peet, and Sharon Selzo for setting up the systems.

References

1. R. Bijlani, Y. Cheng, D. A. Pearce, A. I. Brooks, and M. Ogihara, “Prediction of biologically significant components from microarray data: Independently Consistent Expression Discriminator (ICED),” *Bioinformatics* 18:1-9, 2002
2. T. T. Golub, D. K. Slonim, et al., “Molecular classification of cancer: class discovery and class prediction by gene expression monitoring,” *Science* 286:531-537, 1999
3. http://www-genome.wi.mit.edu/cgi-bin/cancer/datasets.cgi
4. B. Efron, *The jackknife, the bootstrap, and other resampling plans*, Society for Industrial and Applied Mathematics, Philadelphia, Pa, 1982
5. Yeoh E J, Ross ME, Shurtleff SA, Williams WK, Patel D, Mahfouz R, Behm FG, Raimondi SC, Relling MV, Patel A, Cheng C, Campana D, Wilkins D, Zhou X, Li J, Liu H, Pui Ch, Evans WE, Naeve C, Wong L, and Downing JR., “Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling,” *Cancer Cell*, 1(2):109-10, 2002

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Send us your comments in one of the following ways:

- Use the online **Contact us** review redbook form found at:
  ibm.com/redbooks
- Send your comments in an Internet note to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYJ Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	IBM®	POWER4+™
AIX 5L™	ibm.com®	pSeries®
@server™	IntelliStation®	Redbooks(logo) ™
@server™	POWER4™	VisualAge®

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.