Impacts of Multiprocessor Configurations on Workloads in Bioinformatics

Youfeng Wu, Mauricio Breternitz Jr, Victor Ying
Programming Systems Lab
Microprocessor Technology Labs
Intel Corporation
{youfeng.wu,mauricio.breternitz.jr,victor.ying}@intel.com

Abstract

Bioinformatics is among the most active research areas in computer science. In this study, we investigate a suite of workloads in bioinformatics on two multiprocessor systems with different configurations, and examine the effects of the configurations on the performance of the workloads. Our result indicates that the configurations of the multiprocessor systems have significant impact on the performance and scalability of the workloads. For example, a number of workloads have significantly higher scalability on one of the systems, but poorer absolute performance than on the other system. However, traditional scalability failed to capture the impacts of the system configurations on the workloads. We present insights on what kinds of workloads will run faster on which systems and propose new metrics to capture the impacts of multiple processor configurations on the workloads. These findings not only provide an easy way to compare results running on different systems, but also enable re-configuration of the underlying systems to run specific workloads efficiently. We also show how processor mapping and loop spreading may help map the workoads to the underlining multiprocessor configuration and achieve consistent scalability for these workloads.

1 Introduction

Bioinformatics is one of the most active research areas in computer science, and it relies heavily on many types of data-mining techniques. A suite of the well known workloads in bioinformatics includes SNP (Single Nucleotide Polymorphisms), SVM-RFE (Disease Gene Finding in Microarrays), RSEARCH (homologous RNA sequence), SEMPHY (Structural EM Phylogenetic Reconstruction), GENENET (Gene Expression analysis in microarrays), and PLSA (Parallel Linear Space Alignment). The suite of workloads uses such techniques as Bayesian Networks, Classification and Prediction, and Optimization. The workloads and their characteristics are listed in Table 1. The report in [1] introduces and studies the workloads in term of their scalability on a shared memory multiprocessor system (or simply multiprocessor). In this study, we investigate the workloads on two multiprocessor systems with different configurations, and examine the impacts of the configurations on the performance and scalability of the workloads.

Table 1: Bioinformatics workloads and characteristics

Category	Workloads	Applicability
Bayesian Network/ Structure Learning	**SNPs** (Single Nucleotide Polymorphisms)	Pattern recognition Speech recognition Optimization Text mining Game Decision Making
	GeneNet (Gene Expression analysis in microarrays)	
	SEMPHY (Structural EM Phylogenetic Reconstruction)	
Classification and Prediction	**RSEARCH** (homologous RNA sequence)	Recognition Classification Prediction Speech recognition, language parsing
	SVM-RFE (Disease Gene Finding in Microarrays)	Pattern recognition Classification optimization
Optimization	**PLSA** (Parallel Linear Space Alignment)	Pattern recognition, Text mining Association Rule Mining Combinatorial optimization

The two multiprocessor systems we studied have different memory and interconnect configurations. We observe from this study that the scalability measurement is not adequate for comparing a workload running on the two systems. The scalability is measured in term of fixed-size speedup [6]. Namely, for a fixed-size problem, how much faster the problem can be solved with more processors, using the sequential or uniprocessor execution time as the baseline. On the surface, one system delivers noticeably better scalability for a number of workloads than the other system does. However, that system has a significantly less powerful memory subsystem, and the overall performance is much worse. Our investigations successfully classified the

1550-6533/07 $25.00 © 2007 IEEE
DOI 10.1109/SBAC-PAD.2007.30

workloads into CPU bounded, which benefit from more powerful processors and more loosely coupled interconnect, and memory bounded, which perform better on systems with a more powerful memory subsystem and tighter interconnect. Base on the observations, we propose two metrics to capture the impacts of multiple processor configurations on the workloads. The *performance-rated scalability* calculates the scalability to the common base of uniprocessor execution time on one of the multiprocessor systems. The *execution time ratio* directly compares the actual execution times. We also present insights on what kinds of workloads will run faster on which systems, and show how to use processor mapping and loop spreading to achieve consistent scalability. These findings not only provide a mechanism to compare results running on different systems, but also enable reconfiguration of the underlying systems to run specific workloads efficiently.

The rest of the paper is organized as follows. Section 2 provides the configuration information for the two multiprocessor systems studied in this paper. Section 3 presents our experiment results. Section 4 describes the related work, and Section 5 summarizes this paper and points out future research directions.

Table 2. System configurations

IBM xSeries 445	Unisys ES7000
16 2.2Gh Xeon processors grouped into 4 clusters L1 = 8k L2 = 512 k L3 = 2 MB 64M L4 cache per cluster 16G memory total Six PtoP links of L4 snooping bus System bus 400MHz, FS bus 3.2GB/s	16 3.0Gh Xeon processors grouped into 4 clusters L1 = 8k L2 = 512 k L3 = 2 MB 32M L4 per cluster 8G memory total Two 4x4 Crossbars to cache/memory System bus 400MHz, FS bus 3.2GB/s

2 Multiprocessor systems studied

The two multiprocessor system configurations used in this study are list in Table 2, and their system architectures are shown in Figure 1 and Figure 2, respectively. Both systems use the Linux operating system. One of the systems is the IBM Xseries 445 [4], and the other system is the Unisys ES7000 [5]. As illustrated in Table 2, each system consists of four clusters with four processors each for a total 16 processors. Each cluster of 4 processors in the IBM system shares 64M L4 cache which is twice as large as the 32M L4 cache shared by each cluster of 4 processors in the Unisys system. The IBM system has a total of 16G main memory, which is twice as much that the Unisys system has (8G). The IBM system has point-to-point connections across processor clusters, whereas the Unisys system's interconnection network (a crossbar switch) requires a switch crossing to get to memory. Finally, the Unisys system has more powerful processors at 3 GHz, compared to the processors at 2.2 GHz on the IBM system.

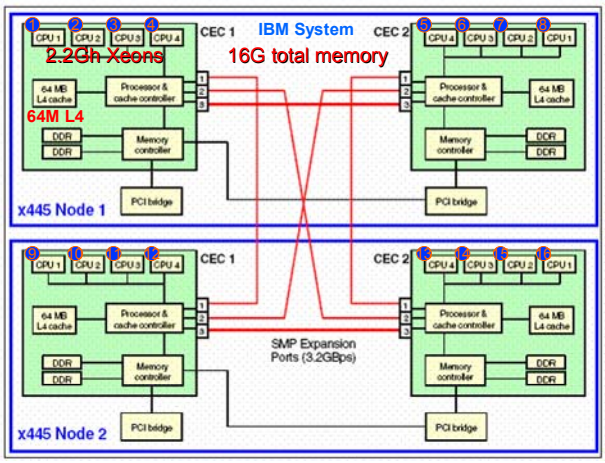

Figure 1. System architecture of the IBM xSeries 445

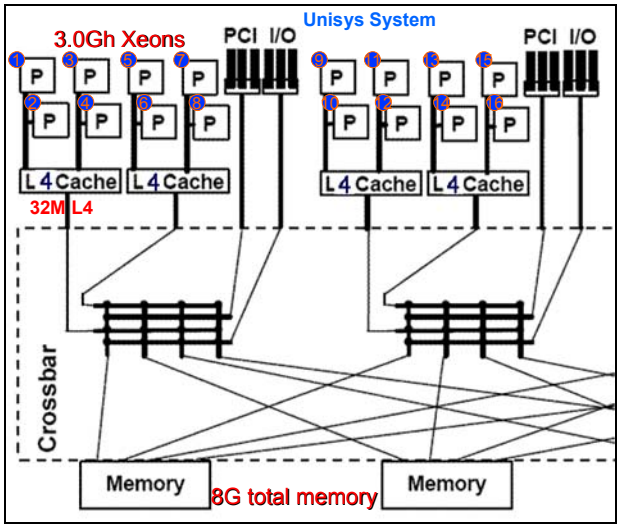

Figure 2. System architecture of the Unisys ES7000

3 Experiment and Results

We use the same set of workloads as reported in [1] in this study. The workloads are hand-parallelized with traditional techniques for locality and load balance, expressed using OpenMP directives. We added processor affinity operations to bind logical processors to physical processors so the performance measurements are more repeatable. For simplicity, we will call the logical processors "threads" and the physical processors

"processors". The workloads are compiled with the Intel IA32 compiler 9.0 at the highest optimization level and linked with Intel's high performance OpenMP library. We run each workload at least three times and use the average execution time as the timing result. Although the Xeon processors in the two multiprocessor systems support hyperthreading, we disable the hyperthreading to focus on multiprocessor aspects of the systems.

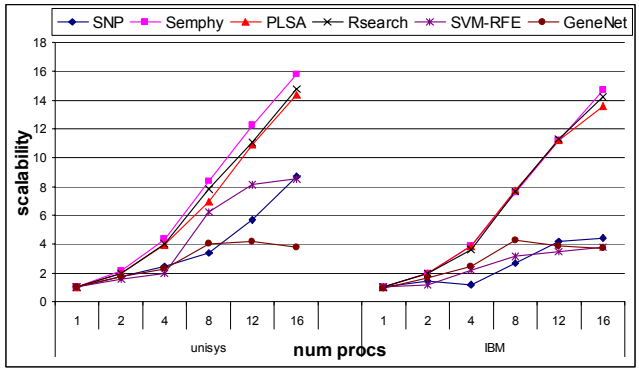

Figure 3. Scalability of the workloads on the two systems

Figure 3 shows the scalability of the workloads on the two systems measured in terms of the fixed-size speedup, namely the ratio of the uniprocessor execution time over the multiprocessor execution time for the fixed input size, with the number of processors ranging from 1 to 16.

Some workloads scaled well on both systems, whereas other presented scaling anomalies, which are investigated and discussed below further.

Three workloads, SEMPHY, PLSA, and RSEARCH, scale very well on both systems, with speedups in the range of 13 to 16 when running with 16 processors. The GENENET workload scales equally poorly on both systems, with a speedup of 4 on 8 and more processors.

The SVM-RFE and SNP workloads, however, show noticeably different scalability on the two systems. For example, when running with 16 processors, SVM-RFE shows a speedup of more than 8 on Unisys system and shows a speedup of less than 4 on the IBM system.

The situation for the SNP workload is similar. Although the IBM system uses slower processors (2.2Gh) than the ones used in the Unisys systems (3.0Gh), the IBM system also has twice as large the L4 cache per cluster and twice as much total main memory as the Unisys system.

As it is not apparent that these differences can explain the difference in scalabilities, we conduct several more experiments to understand the differences in the next few subsections.

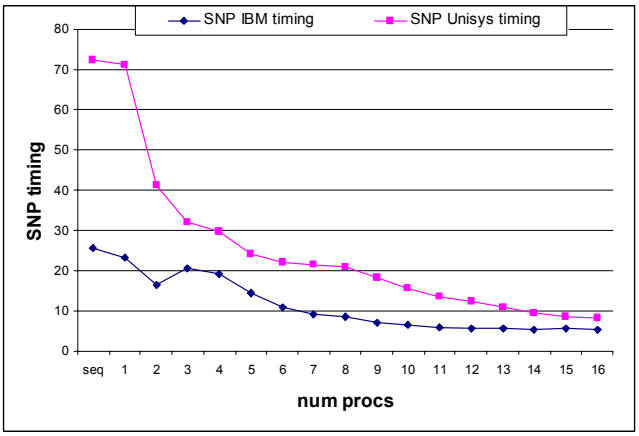

Figure 4. Timing of SNP on the two systems

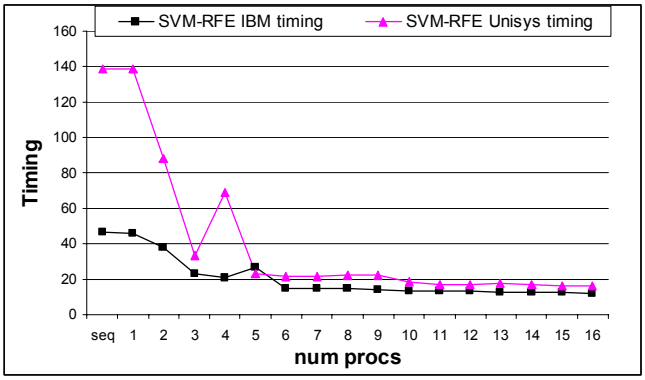

Figure 5. Timing of SVM-RFE on the two systems

3.1 Execution time difference

Figure 4 and Figure 5 show the wall clock timings of the SNP and SVM-RFE workloads on both the IBM and the Unisys systems. Although the SNP and SVM-RFE workloads show less scalability on the IBM system than on the Unisys system, they almost always run faster on the IBM system than on the Unisys system. The timing differences are more significant at uniprocessor (about 2.9 times faster for SNP and 3 times for SVM-RFE) than at more processors (e.g. 1.6 times faster for SNP and 1.4 times for SVM-RFE at 16 processors). For comparison, we plot the execution timing of the RSEARCH workload in Figure 6, and it shows that the RSEARCH workload runs almost always faster on the Unisys system than on the IBM system, even though it shows similar scalability on both systems. Although the timing result indicates that it is usually easier to achieve better scalability if the uniprocessor version runs slowly, it is interesting to understand why the SNP and SVM-RFE workloads run faster on the IBM system while the RSEARCH workload runs faster on the Unisys system.

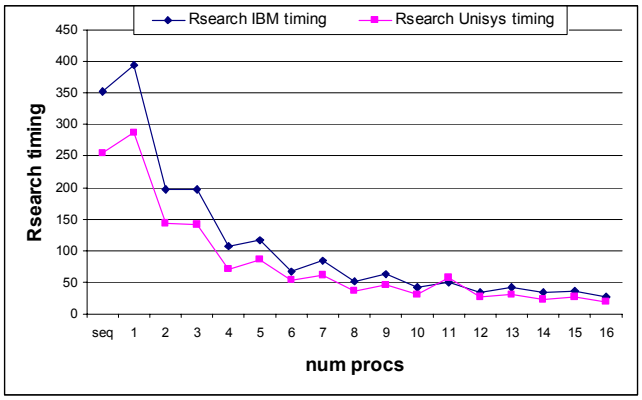

Figure 6. Timing of RSEARCH on the two systems

3.2 Memory bounded and CPU bounded workloads

Since the execution time of a program is usually correlated to the amount of memory stalls and CPU utilization, the timing difference between the workloads can be relatively easily explained by the configuration differences between the two machines. For example, **Figure 7** shows the resident set size (RSS) for the SNP, SVM-RFE, and RSEARCH workloads collected using the Linux "top" command. SNP, SVM-RFE workloads have significantly larger RSS and therefore they may need more memory to run efficiently. The large L4 caches and main memory on IBM system allows the workloads to run much faster on the IBM system than on the Unisys system. On the other hand, the RSEARCH workload has significantly smaller RSS and may be treated as CPU bounded, and it benefits from the faster CPU clock frequency of the Unisys system.

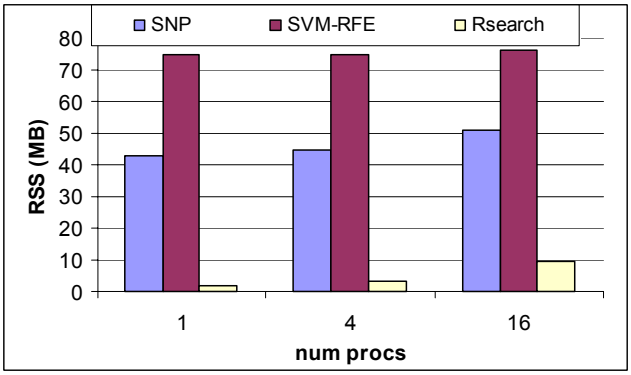

Figure 7. Footprint comparison of three workloads

Having a large RSS size does not necessarily mean that a workload is memory bounded. Further, we measure the instructions retired per cycle (IPC, which gages the CPU utilization) and detailed memory characteristics to demonstrate the memory and CPU boundedness of the workloads. **Figure 8** shows the IPC, front-side bus utilization and bandwidth, and L2/L3 cache miss rates of SNP and RSEARCH workloads when running with four processors, collected with the Intel's VTUNE tool. The RSEARCH workload has twice as high an IPC as the SNP workload (2.10 vs. 0.96). On the other hand, SNP has significant front-side bus (FSB) activity while RSEARCH does not have much, and SNP has significant L2 and L3 cache misses and RSEARCH suffers very little L2 and L3 cache misses. This result and the RSS data reasonably convince us that SNP and SVM-RFE are more memory bounded while RSEARCH is more CPU bounded. In the next subsection, we further explain exactly how the large L4 cache on IBM system can benefit the memory bounded workloads.

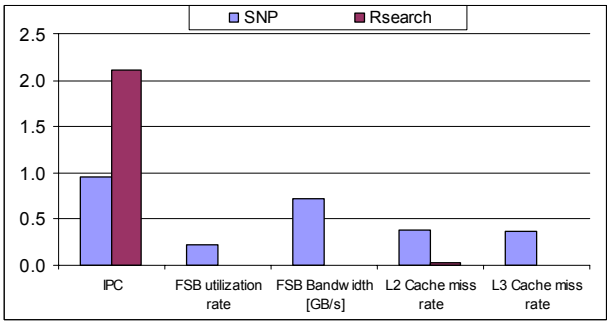

Figure 8. IPC and memory characteristics of SNP and RSEARCH workloads

3.3 L4 cache contentions

The SNP and SVM-RFE workloads have significant memory activities, and the larger shared L4 caches in the IBM machine will accommodate the activities better than the smaller L4 caches in the Unisys system. As the L4 cache is off the processor chip and its activity cannot be measured with the VTUNE tool, we measure the effects of L4 cache on the performance of the workloads. Figure 9 shows what happens when the SVM-RFE and the RSEARCH workloads are parallelized to run on two processors sharing the same L4 cache or using two different L4 caches on both IBM and Unisys systems. The 64M L4 cache on IBM machine accommodates the two processors from SVM-RFE workload quite well, showing only slightly lower performance compared to running the workload on two processors utilizing two different L4 caches. The 32M L4 cache on the Unisys system however is too small to be shared by the two processors for running the SVM-RFE workload, and running on the two processors sharing the same L4 cache is twice as slower as running on the two processors utilizing two different L4 caches. The RSEARCH workload, on the other hand, is not memory bounded and can be accommodated well with the 32M L4 cache, and therefore there is not much performance difference to run on two processors either sharing the same L4 cache or

utilizing two different L4 caches. These results further demonstrate that the IBM system has more robust memory system and can run memory bounded workloads better than the Unisys system.

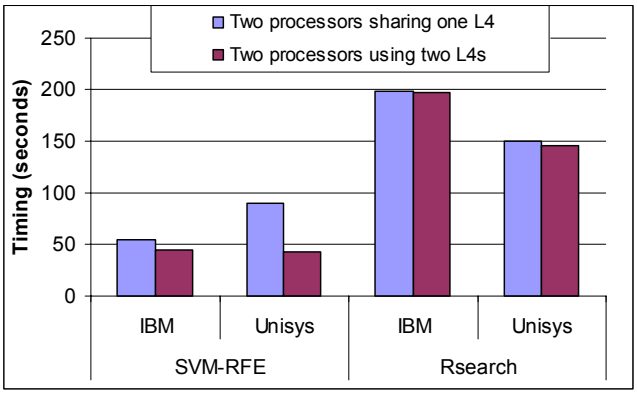

Figure 9. Map two threads to two processors using the same L4 cache or different L4 caches

3.4 Performance-rated scalability

Although we now understand the system configurations have significant impact on the execution time and scalability of the workloads, the fixed-size scalability measurement (e.g. the measurement in Figure 3) is unable to compare the performance of the workloads on different systems. In order to capture the impacts of different systems on workloads, we define the "*performance-rated scalability*" as the ratio of uniprocessor execution time on one of the systems (e.g. the Unisys system) over the parallel execution times on every other system (e.g. the IBM systems). The performance-rated scalability (denoted perf+scalability in the figures) for SVM-RFE and RSEARCH is shown in Figure 10 and Figure 11, respectively. The IBM system has better performance-rated scalability than the Unisys system for any number of processors for the SVM-RFE workload, while the Unisys system has better performance-rated scalability than the IBM system for RSEARCH workload, across various numbers of processors. The differences in performance-rated scalability capture the ability of the Unisys system to run CPU bounded applications, such as RSEARCH, faster than the IBM system, and also the ability of the IBM system to run memory bounded applications, such as SVM-RFE, better than the Unisys system. We believe these pictures give more accurate qualifications of the relative performance and scalability for the workloads on the two different multiprocessor systems than the fixed-size scalability shown in Figure 3.

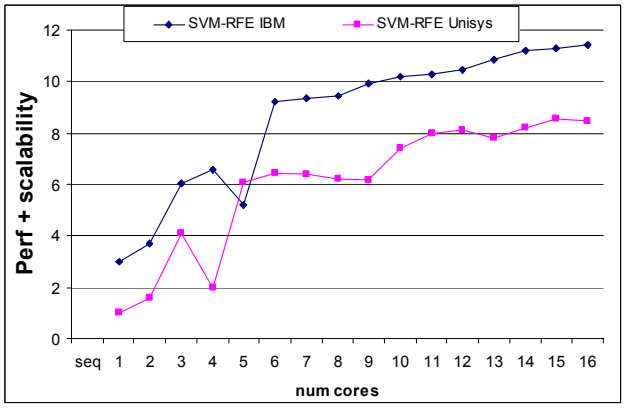

Figure 10. Scalability of SVM-FRE with the uniprocessor Unisys time as the base

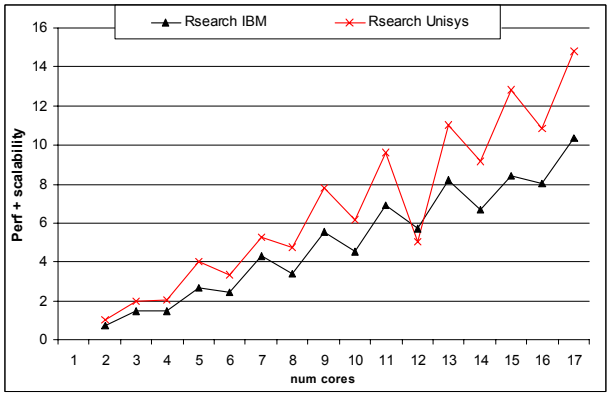

Figure 11. Scalability of RSEARCH with the uniprocessor Unisys time as the base

3.5 Execution time ratio scalability

Another way to relate execution time to scalability is to compute the ratio of the execution time on the Unisys and that on the IBM system at each number of processors. For example, if for a workload the ratio of Unisys execution time over the IBM execution time gets smaller as the number of processors increase, the scalability of workloads on the Unisys should be better than the IBM system, as in this case the Unisys system is able to utilize the multiprocessors resource to improve execution speed better than the IBM system does. If we draw the execution time ratios in a line graph, a horizontal line would indicate equal scalability, while an increasing line shows that the IBM system scales better and a decreasing line shows that the Unisys system scales better.

Figure 12 shows the execution time ratios for SVM-RFE, SNP, and RSEARCH workloads from 1 to 16 processors, with the ratio at the uniprocessor normalized to 1. From this figure, we can see clearly that the RSEARCH workload scales similarly on the two systems as the ratios for RSEARCH is mostly on a flat line. The

SVM-RFE and SNP workloads scale better on Unisys system mostly from 1 processor to 3 processors, since the ratios decrease sharply from 1 to 3 processors, and do not scale much better on the Unisys system than on the IBM system with 5 or more processors, as the two lines are relative flat in that range. It is interesting to note that, although the result in Figure 3 shows that the SVM-RFE workload scales significantly better on the Unisys system than on the IBM system with 8 or more processors, this is not really the case according to Figure 12, as the time ratios do not change much from 8 to 16 processors. It looks like that the traditional speedup measurement exaggerates the speed improvement over the uniprocessor execution time while the execution time ratios capture the execution time differences at the individual numbers of processors.

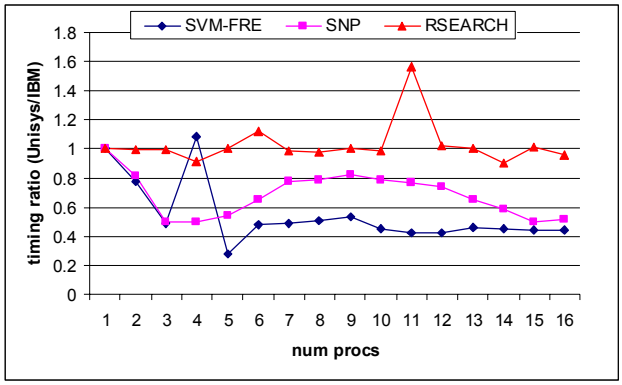

Figure 12. Ratios of execution times on Unisys system over that on IBM system

3.6 Processor mapping effect

The performance of the workloads can also be indirectly impacted by the processor mapping algorithms that may determine the fitness of the workloads to the multiprocessor configuration. For example, in Figure 10, there are noticeable performance dips for the SVM-RFE workload when the number of processors increases from 4 to 5 on the IBM system (the perf+scalability for SVM-RFE drops from 6.6 to 5.2) and when the number of processors increases from 3 to 4 on the Unisys system (the perf+scalability for SVM-RFE drops from 4.1 to 2.0). These significant performance losses when more processors are used clearly indicate the need to properly match workloads with processor configuration. Specifically, our experiment sequentially maps thread *i* to processor *i* via processor affinity. This method maps adjacent threads to processors that share the L4 caches as much as possible and gives better performance than many other mapping methods on the average. However, this strategy does not work well in a few situations. For the SVM-RFE workload running with 4 threads on 4 processors on the IBM system, the sequential mapping uses processors 1, 2, 3, and 4 for the four threads, which all share the same L4 cache. When the workload runs with 5 threads on 5 processors, the 5th processor uses a different L4 cache. Unfortunately, thread 5 needs to share data with thread 4, and the sharing has to go through the cache coherency mechanism, which is slow. If we adjust the processor mapping to place threads 4 and 5 on processors 5 and 6 so they can share the same L4, the performance-rated scalability immediately increases from 5.2 to 7.4, as shown in Figure 13. For the SVM-RFE workload running with 3 processors on the Unisys system, the sequential mapping uses processors 1, 2, and 3 for the 3 threads, which share the same 32M L4 cache. When the workload runs with 4 processors to share the same L4 cache, the L4 cache cannot accommodate the additional demand, and the performance drops by more than 50%. If we adjust the processor mapping to place threads 3 and 4 on processors 5 and 6 so they can access a different L4 cache, the performance rated scalability increases from 2 to 5.1, as illustrated in Figure 13.

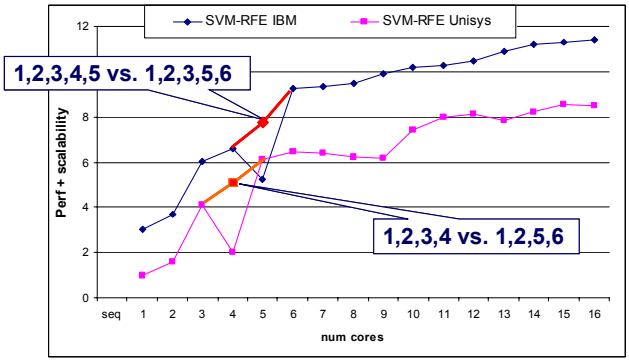

Figure 13. Effects of processor mapping on performance and scalability

This experiment suggests that processor mapping may significantly affect performance. The optimal processor mapping depends not only on the multiprocessor configuration, but also on the workload characteristics, e.g. how big is the work set and how the data is shared. Determining the optimal mapping through program analysis or profiling seems quite challenging. It may also not be practical to experiment with all possible processor mappings, especially when the number of processors is large, e.g. >=16. However, due to the regularity of the system configuration, the search space is much constrained. For example, for the IBM system, the processors inside the same cluster are symmetric to each other, and the different clusters are similar to each other. To find the optimal mapping of the 4 threads to 4 of the 16 processors, we only need to experiment with five mappings listed in Figure 14. This number is significantly smaller than the $\binom{16}{4}$ possibilities if we were to search for the optimal mapping exhaustively. Note that, one of the five mappings in

Figure 14 is the best mapping (1,2,5,6) for the SVM-RFE workload with 4 threads on the Unisys system.

1	2	3	4												
1	2	3		5											
1	2			5	6										
1	2			5				9							
1				5				9				13			

Figure 14. Searching for the best mapping of 4 threads to 4 of the 16 processors

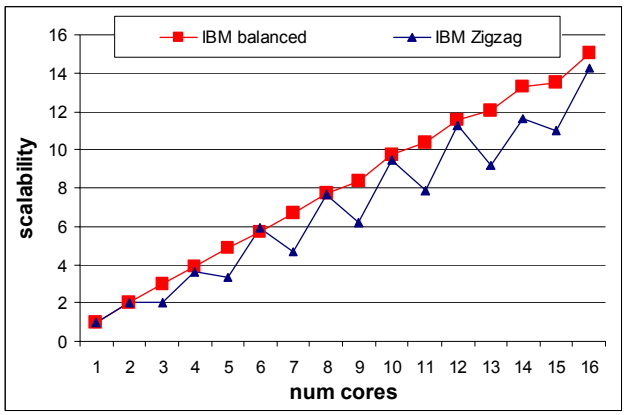

Figure 15. Effects of processor load balance on scalability

3.7 Task partitioning effect

Task partitioning can also affect whether or not the program execution fits well with the multiprocessor configuration and thus may impact the performance of the workloads on the systems. Figure 11 shows a zigzag anomaly for the RSEARCH workload, where performance drops whenever the number of processors increases from an even number to an odd number. The zigzag picture for RSEARCH workload on the IBM system is redrawn in Figure 15 (see the line marked with "IBM zigzag"). This anomaly is tracked down to a task partitioning issue in the algorithm. For correctness, the algorithm has to partition the problem into an even number of tasks. When the number of processors, P, is an odd number, the algorithm partitions the problem into even number (P+1) tasks, causing load unbalancing as the P+1 tasks cannot be evenly distributed on the P processors. However, we may partition the problem into 2*P tasks so that they can be evenly distributed onto the P processors. This solution works well to achieve load balance for any number of processors, as shown in Figure 15 (see the line marked with "IBM balanced").

3.8 Discussion

The above experiments show that the workloads exhibit a wide spectrum of CPU and memory boundedness, and can benefit from balanced multiprocessor systems with fast CPUs and efficient memory subsystems. Since CPU cost is only a small portion of the overall multiprocessor system cost, employing fast CPU clock frequency is cost effective to help those CPU-bound workloads. Some workloads are very sensitive to cache configuration and processor structure, and careful analysis and tuning of the workloads, such as processor mapping and loop spreading are often necessary to efficiently utilize the available processor resource.

4 Related work

Bioinformatics represents the emerging applications that are being studied actively [1, 2, 14]. The study in [1] reported the scalability results of the workloads on one multiprocessor system. We use the same set of workloads in bioinformatics as in [1], and extended the study to compare the scalability and performance of the workloads on two multiprocessor machines with different configurations.

This paper starts with the desire to understand the scalability differences of the bioinformatics workloads on the two different multiprocessor machines. This is related to the topic of how to define speedup, which is richly discussed and published in the literature. For example, [6] discussed three types of speedup metrics: fixed size speedup, fixed-time speedup, and memory-bounded speedup. Fixed-size speedup emphasizes how much execution time of a fixed size problem can be reduced with parallel processing. This speedup is often used to measure the scalability of existing workloads on cost-effective systems. Amdahl's law [10] is based on the fixed-size speedup. Both fixed-time and memory bounded speedups are scaled speedups, which are concentrated on exploring the computational power of parallel computers for solving otherwise intractable large problems by scaling the problem size when more processing power is used. As the number of processors increases, fixed-time speedup scales problem size to meet the fixed execution time. Then the scaled problem is also solved on a uniprocessor to get the speedup. As the number of processors increases, memory-bounded speedup scales problem size to utilize the associated memory increase. Fixed size speedup usually is smaller than the scaled speedups, as 1) the sequential bottleneck is more detrimental to a fixed problem size when the number of processors increases; and 2) the communication and synchronization overheads increase with the number of processors and the problem size needs to increase in order to maintain a constant communication/computation ratio.

While scalability has been widely used as an important property in analyzing algorithms and architectures, execution time is the dominant metric of parallel processing [7]. Scalability studies would have

little practical impact if they could not provide useful information on time variation in a scalable computing environment. Sun et al. studied the relation between execution time and scalability in [7], defined a scaled scalability called the isospeed scalability (the average unit speeds with scaled workload on multiprocessors), and proved that for any pair of algorithm-machine combinations (AMC) which have the same initial execution time, an AMC has a smaller scalability if and only if it has a larger execution time on scaled problems. In other words, the same scalability will lead to the same execution time, and vice versa.

The study in this paper initially uses the fixed-size speedup as the scalability measurement as our problem sizes cannot change. This scalability measurement is not able to compare the performance (execution time) of a workload on different architectures. For example, with fixed size speedup, we will not be able scale the problem size to obtain the same initial execution time as required by [7] for meaningful measurement. We propose the performance-rated scalability to relate execution time with scalability for our fixed size situation, and it provides a direct comparison of performance and scalability of the workloads in the two multiprocessor systems. We also use the execution time ratios to compare the scalabilities of the workloads on the two systems. This metric is motivated by, but distinct from the isospeed scalability [7].

A memory-bounded workload is one whose execution time is dominated by the time spent accessing memory [11]. Parallel processing on distributed systems has traditionally paid close attention to the memory boundedness of the workloads and managing memory operations explicitly to reduce communication overhead and improve the overall performance [8, 9]. For shared memory systems the memory subsystem is often indirectly managed with such transformations as tiling and data layout [12, 13]. In this study, we demonstrate that the performance of the workloads can suffer significantly on multiprocessor systems if the memory operations are not properly managed. Specifically, we showed that proper processor mapping can significantly improve memory locality, reduce cache contentions, and enhance overall parallelism.

The idea of dividing tasks into smaller pieces so they may be more evenly distributed on the available processors was studied in [3]. The similar idea proved useful to improve the load balance of the RSEARCH workload studied in this paper.

5 Summary and future work

In this paper we experiment with a suite of workloads in bioinformatics on two multiprocessor systems with different memory and interconnect configurations. We observe that the traditional scalability measurement is not adequate for comparing a workload running on the two systems. On the surface, one system shows noticeably better scalability than the other for a few workloads. However, that system has a significantly less powerful memory subsystem, and the actual performance is much worse than the other for the workloads. We propose performance-rated and execution time ratio scalabilities to capture the performance and scalability differences. We present insight on what kinds of workloads will run faster on which systems. We also show how processor mapping and loop spreading may help achieving more consistent scalability.

Several areas are important for future investigations. First, we should continue the effort to extend the fixed-size scalability measurement to a more useful measurement so it can tell the performance (execution time) variations on different multiprocessor systems even when the workloads have fixed input sizes. Our performance-rated scalability and time ratios are the first step along this direction. Second, we may continue investigations of memory locality and processor mapping issues for the workloads on the shared multiprocessor systems. Even though the communication and synchronization are mostly managed by hardware in shared multiprocessor systems, proper data locality transformation and processor mapping can significantly improve performance. A unified approach that combines memory locality transformations with processor mapping seems in interesting direction to pursue.

6 Acknowledgements

We would like to thank Jesse Fang for his support, and many colleagues, including Jin Liu, Yongjian Chen, Jack Liu, Tin-Fook Ngai, Dong-Yuan Chen, Uma Srinivasan, Vijay Menon, Brian Murphy, for their contributions to this work.

7 References

[1] Y. Chen, Q. Diao, C. Dulong, C. Lai, W. Hu, E. Li, W. Li, T. Wang, Y. Zhang, "Performance Scalability of Data-Mining Workloads in Bioinformatics." Intel Technology Journal. http://developer.intel.com/technology/itj/2005/volume09issue02/ art04_data_workloads/p01_abstract.htm (May 2005)

[2] Pradeep Dubey, “Recognition, Mining and Synthesis Moves Computers to the Era of Tera”, Intel Technology Journal, http://www.intel.com/technology/magazine /computing /recognition-mining-synthesis-0205.htm (February 2005)

[3] Y. Wu and T. Lewis, "Parallel Processor Load Balance through Loop Spreading," Proc. Supercomputing '9, Nov. 1989.

[4] IBM Corp, “Intel processor-based servers: x445”, http://www-03.ibm.com/servers /eserver/xseries/x445.html

[5] Unisys Inc., “Intel® Xeon Processor multiprocessor 32-bit computing on ES7000 Real-Time Enterprise Servers,” http://www.unisys.com/products/es7000__servers/hardware/ xeon__processor__mp.htm

[6] Xian-He Sun; Rover, D.T.; "Scalability of parallel algorithm-machine combinations," IEEE Transactions on Parallel and Distributed Systems, Volume 5, Issue 6, June 1994 Page(s):599 - 613

[7] Xian-He Sun; "The relation of scalability and execution time," Proceedings of IPPS '96, The 10th International Parallel Processing Symposium, 1996, 15-19 April 1996 Page(s):457 - 462

[8] Ching-Hsien Hsu; Yeh-Ching Chung; Don-Lin Yang; Chyi-Ren Dow; "A generalized processor mapping technique for array redistribution", IEEE Transactions on Parallel and Distributed Systems, Volume 12, Issue 7, July 2001 Page(s):743 - 757

[9] E.T. Kalns and L.M. Ni, " Processor mapping techniques toward efficient data redistribution," IEEE Transactions on Parallel and Distributed Systems, Volume 6, Issue 12, Dec. 1995 Page(s):1234 - 1247

[10] G. Amdahl, "Validity of the single-processor approach to achieving large scale computing capabilities," Proc. AFIPS Conf, pp. 483485, 1967.

[11] Xiaodong Zhang; Yanxia Qu; Li Xiao, "Improving distributed workload performance by sharing both CPU and memory resources", Proceedings. 20th International Conference on Distributed Computing Systems, 2000. 10-13 April 2000 Page(s):233 - 241

[12] S. Coleman and K.S. McKinley, "Tile Size Selection Using Cache Organization and Data Layout," Programming Language Design and Implementation, June 1995.

[13] K. Hogstedt, L. Carter, J. Ferrante, "On the parallel execution time of tiled loops," IEEE Transactions on Parallel and Distributed Systems, Volume 14, Issue 3, March 2003 Page(s):307 – 321

[14] D.A. Bader, Y. Li et al., "BioPerf: a benchmark suite to evaluate high-performance computer architecture on bioinformatics applications," IEEE International Workload Characterization Symposium:163-73, 2005.