

# Scalability of SPEC OMP on Fujitsu PRIMEPOWER

*Naoki Sueyasu, Hidetoshi Iwashita<sup>1</sup>, and Kohichiro Hotta<sup>1</sup>*  
*Strategy and Technology Division, Software Group*  
*Fujitsu Ltd.*  
*140 Miyamoto*  
*Numazu-shi, Shizuoka 410-0396, Japan*

*Matthijs van Waveren*  
*Fujitsu Systems Europe*  
*8 rue Maryse Hilsz*  
*Parc de la Plaine*  
*31500 Toulouse, France*

*Kenichi Miura*  
*Computer Systems Group*  
*Fujitsu Ltd.*  
*4-1-1 Kamikodanaka, Nakahara-ku,*  
*Kawasaki-shi, Kanagawa 211-8588, Japan*

## Abstract

The SPEC OMPM2001 and OMPL2001 suites can be used as a yardstick for measuring and comparing shared-memory systems, which support the OpenMP API. We investigate the scalability of selected components in the large dataset version of the benchmark suite, and compare the results with the scalability of the corresponding components of the medium dataset version of the benchmark suite. We have run the described components of the SPEC OMP benchmark on the soon to be released Fujitsu PRIMEPOWER HPC2500 system. We analyze the benchmark results with respect to scalability and load balance. The scalability of at least one of the components has improved in the large dataset version.

## 1. Introduction

The large dataset version of the SPEC OMP benchmark suite has been released in May 2002, and the medium dataset version in June 2001. The large dataset version of the suite, SPEC OMPL2001, uses working sets of up to 6.5 GB. This compares with datasets of 1.6 GB for the medium dataset version, SPEC OMPM2001, and of 200 MB for SPEC CPU2000.

The scalability of the different components of the medium dataset version of the suite is described in [1] and [2]. It is shown that the components can be subdivided into four categories: components which show superlinear scalability, components which show good scalability for all number of processors, components which show good scalability up to 64 processors, and components which show poor scalability. The benchmark suite is thus a reasonable representation of realistic applications running on a computer centre system.

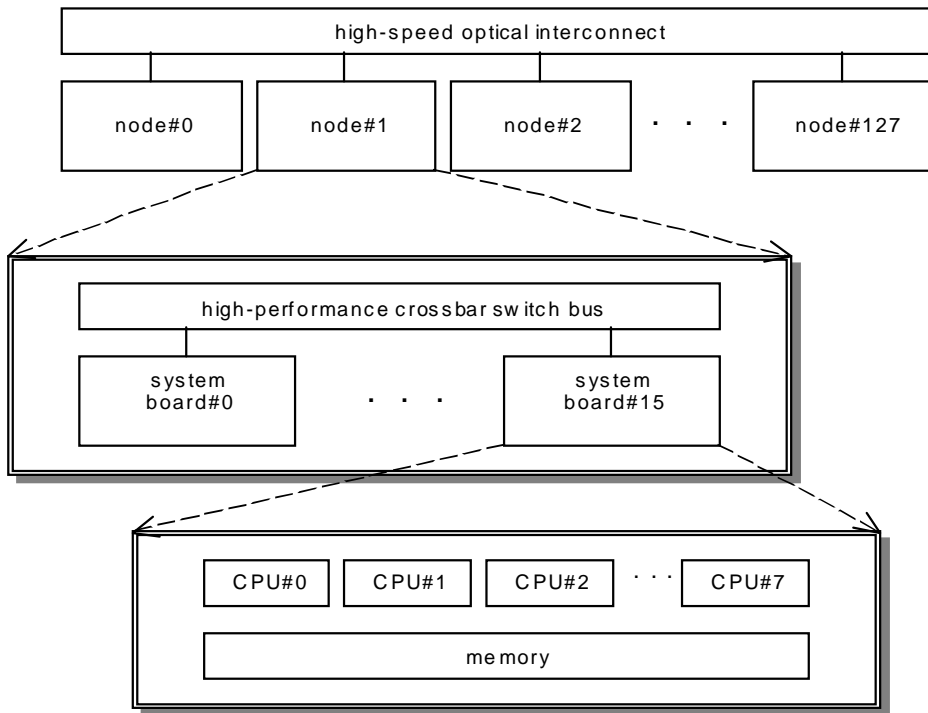
---

<sup>1</sup> Advanced Parallelizing Compiler Project

In this paper we discuss the medium dataset size and the large dataset size versions of the SPEC OMP benchmark suite. We investigate the scalability of selected components in the large dataset version of the benchmark suite, and compare the results with the scalability of the corresponding components of the medium dataset version of the benchmark suite. We have run the described components of the SPEC OMP benchmark on the soon to be released Fujitsu PRIMEPOWER HPC2500 system. This system is a flat SMP system with up to 128 high-performance SPARC64 V processors, which can be deployed in stand-alone or multi-server clusters. We have used the Parallelnavi software environment, which contains compilers for the OpenMP Fortran 2.0 [5], and OpenMP C and C++ 2.0 language specifications [6].

## 2. Description of Hardware

The Fujitsu PRIMEPOWER HPC2500 is a parallel computation server supporting up to 128 CPUs and 512 gigabytes of memory per node. The CPU used is SPARC64 V (1.3 GHz), which conforms to the SPARC International V9 architecture. Figure 1 shows the maximum configuration of the system. Each node has 16 system boards and each system board has 8 CPUs, and 32 gigabytes of memory. All system boards within nodes are connected with the crossbar switch. The nodes are connected with the optical interconnect and the maximum configuration of the system is 128 nodes.

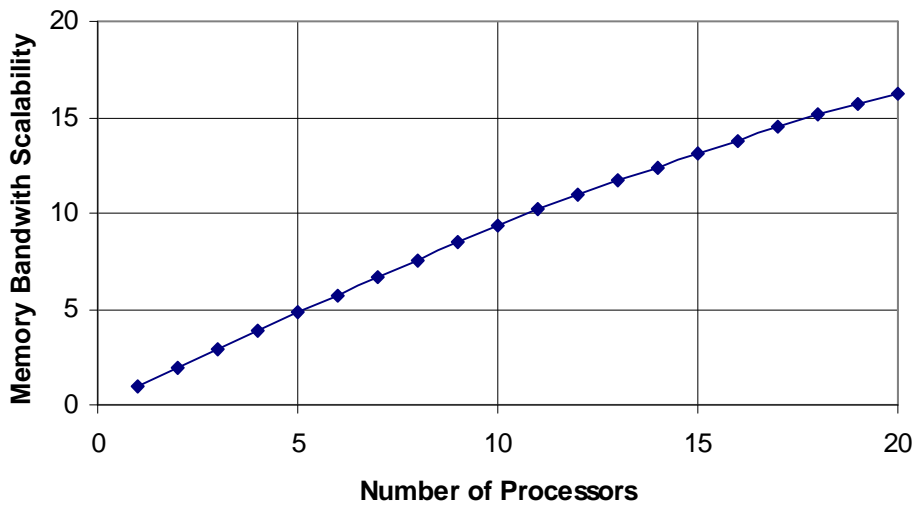


**Figure 1: PRIMEPOWER 2500 System**

### 2.1 High Memory Bandwidth

The Fujitsu PRIMEPOWER HPC2500 has a very high memory bandwidth. The system bus which is structured by high performance crossbar switch has the performance of 133 GB/s.

Figure 2 shows the scalability of the STREAM benchmark of this system. It shows that the memory bandwidth is very scalable.

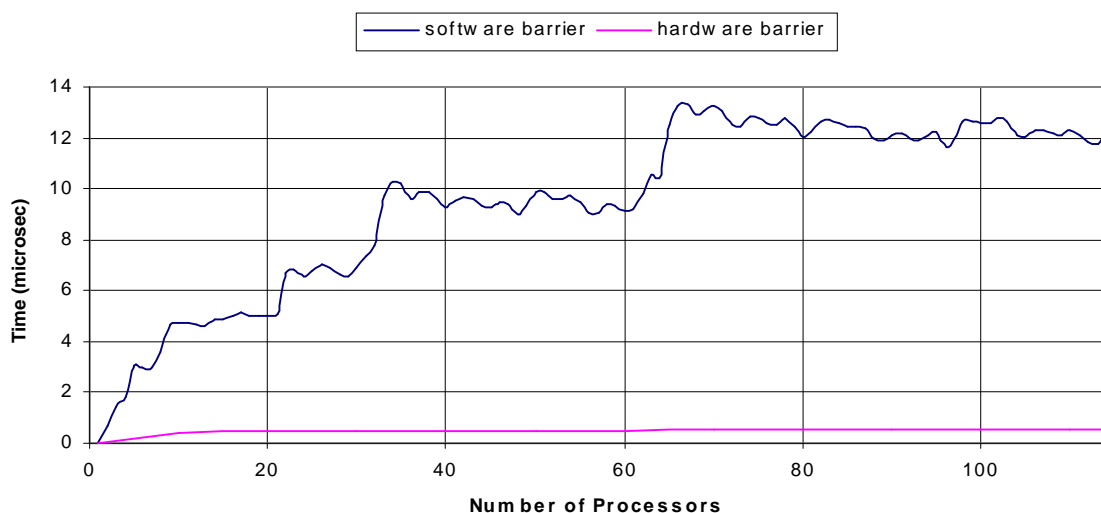


**Figure 2: Scalability of the STREAM benchmark**

## 2.2 Hardware Barrier Facility

This system has a hardware barrier facility that reduces the execution cost of the barrier synchronization between threads.

There are two implementations of the OMP BARRIER directive. One utilises a software barrier, and the second utilises an hardware barrier. The timings of both implementations are shown in Figure 3. We see that the hardware barrier implementation is an order of magnitude more efficient than the software barrier implementation. The timing of the software barrier implementation shows jumps at even powers of 2 in the number of processors. This is due to the fact that extra operations in the form of a stage shuffle are added when the number of processors reaches  $2^n+1$ . The effect of the hardware barrier is seen very clearly when running the SPEC OMP benchmark GALGEL. There is a 10 % improvement of performance with the hardware barrier.



**Figure 3: Timings of the OMP BARRIER directive, as implemented as a software barrier, and as an hardware barrier**

### 3. System Software

Parallelnavi is a software package for the PRIMEPOWER series server, and includes Fortran and C/C++ compilers, support tools, mathematical libraries, a message passing library (MPI), and a multifunctional job management environment. Parallelnavi Fortran V2.1 supports the OpenMP Fortran API Version 2 [5]. Parallelnavi C/C++ V2.1 will support the OpenMP C/C++ API Version 2.0 [6].

While OpenMP Fortran 1.1 was aligned with Fortran 77, OpenMP Fortran 2.0 has been updated to be aligned with Fortran 95. While OpenMP C/C++ 1.0 was aligned with OpenMP Fortran 1.0 and C89, OpenMP C/C++ 2.0 has been updated to the level of C99 and OpenMP Fortran 2.0.

#### 3.1 OpenMP API V2.0 Compiler Support

Our latest compiler supports OpenMP API V2.0, which includes the following major features:

- WORKSHARE construct,
- NUM\_THREADS clause,
- COPYPRIVATE clause, and
- Array REDUCTION clause.

The NUM\_THREADS clause, which specifies the number of threads to execute the parallel region, was implemented with the same functionality as the OMP\_SET\_NUM\_THREADS library call. The COPYPRIVATE clause was implemented to make the execution of the threads keep the following order: 1) the SINGLE thread specified with the COPYPRIVATE clause writes out the value of the variable to a global temporary, 2) all threads perform barrier synchronization, and 3) every thread reads the global temporary into its own private variable. Our implementation of the array REDUCTION is a simple and natural expansion of the scalar REDUCTION in the OpenMP API V1.1.

#### 3.2 CPU Grouping and Thread Binding

In existing usual Unix systems, running jobs are affected when high-load TSS processing or another job is executed or an I/O interrupt occurs. In this case, the job that is running is affected in the way that resources that the job requires may not be allocated or processing of the job may be suspended. In such systems, execution time cannot be guaranteed because of time variations resulting from the system state.

Parallelnavi allows to prevent processing other than job processing from affecting the job execution by grouping the CPUs of a Node into those that execute jobs and those that perform other processing (TSS processing and system processing such as daemon processing) and managing these groups separately. When the grouping function is used, I/O interrupts do not occur on a CPU that executes jobs. Job execution is therefore not suspended by I/O interrupts. In Parallelnavi, a CPU assigned to execute jobs is called a CPU for jobs and a CPU assigned for performing other processing is called a CPU for TSS. A CPU for TSS can also be specified so that that CPU would not receive any I/O interrupts. By estimating the amount of CPU resources required by jobs and setting up a CPU configuration suitable for the estimate, CPUs for jobs can always be obtained and the job execution time guaranteed.

### 4. SPEC OMP Benchmark Suites

There are two SPEC OMP benchmark suites, viz. the OMPM2001 suite with medium dataset sizes, and the OMPL2001 suite with large dataset sizes. The benchmark codes in both suites originate from the SPEC CPU2000 benchmark suite, and they have been parallelised with OpenMP. OpenMP was chosen, because the benchmarks are focused on SMP systems. OpenMP offers an industry-wide standard method of SMP parallelisation for Fortran, C, and C++. Since it is directive based, it allows the serial version to remain largely intact. It also offers a quick path to parallel code conversion.

A comparison of the SPEC OMP suites and the SPEC CPU suite is shown in Table 1. The OMP suites have larger working set sizes than the CPU suite, viz. 6.5 GB for OMPL and 1.6 GB for OMPM versus 200 MB for CPU. The OMP suites also have longer runtimes, viz.  $O(10^4)$  sec for OMPM vs.  $O(10^3)$  sec for the CPU suite.

In order to make the running of the benchmark suites as simple and user-friendly as possible, the same tools are used to run all 3 benchmark suites, the run and reporting rules are similar, and the output format of the results is similar. The geometric mean is used in all suites to compute the overall performance relative to a baseline system. The latest results can be viewed on the SPEC web site: <http://www.spec.org/hpg/omp2001>.

| Characteristic       | CPU2000          | OMPM2001                | OMPL2001                |
|----------------------|------------------|-------------------------|-------------------------|
| Maximum working set  | 200 MB           | 1.6 GB                  | 6.5 GB                  |
| Memory Needed        | 256 MB           | 2 GB                    | 8 GB                    |
| Benchmark runtime    | 30 min @ 300 MHz | 5 hrs @ 300 MHz         | 9 hrs @ 300 MHz         |
| Language             | C, C++, F77, F90 | C or C++, F90, OpenMP   | C or C++, F90, OpenMP   |
| Focus                | Single CPU       | < 16 CPU system         | > 16 CPU system         |
| System Type          | Cheap desktop    | MP workstation          | Engineering MP system   |
| Runtime              | 24 hours         | 34 hours                | 75 hours                |
| Runtime 1 CPU        | 24 hours         | 140 hours               | 1000 hours              |
| Run modes            | Single and rate  | Parallel                | Parallel                |
| Number of benchmarks | 26               | 11                      | 9                       |
| Iterations           | Median 3 or more | Worst of 2, median of 3 | Worst of 2, median of 3 |
| Source mods          | Not allowed      | Allowed                 | Allowed                 |
| Baseline flags       | Max of 4         | Any, same for all       | Any, same for all       |
| Reference system     | 1 CPU @ 300 MHz  | 4 CPU @ 350 MHz         | 16 CPU @ 300 MHz        |

**Table 1 : Comparison of Benchmark Suites**

## 5. Scalability Analysis

The scalability of the different components of the medium dataset version of the suite is described in [1] and [2]. It is shown that the components can be subdivided into four categories: components which show superlinear scalability, components which show good scalability for all number of processors, components which show good scalability up to 64 processors, and components which show poor scalability. In order to compare the scalability of the medium and the large dataset versions of the suites, we focus on the WUPWISE and the SWIM benchmark which show good scalability, on the MGRID benchmark which shows good scalability up to 64 processors, and on the FMA3D benchmark which shows poor scalability. We have performed measurements of the performance of these components on a Fujitsu PRIMEPOWER HPC2500 with 32, 64, and 96 CPUs. We used the Fujitsu OpenMP Fortran compiler.

### 5.1 WUPWISE

The Wuppertal Wilson Fermion Solver (wupwise) is a program in the field of lattice gauge theory. It solves the inhomogeneous lattice-Dirac equation via the BiCGStab iterative method which has established itself as the method of choice. In [1] and [2], it is shown that this component of the medium suite, SPEC OMPM2001, falls in the category of applications that scale well for all number of processors. The reasons are that the main computation is a matrix vector multiply (zgemm), which parallelizes well, and that the outer loop is parallelized.

In this study, we have measured the performance of this benchmark component in the large suite, SPEC OMPL2001. Figure 4 shows the scalability of this component in both the medium and the large benchmark suites for 32, 64, and 96 CPUs. We see that WUPWISE scales well in both suites. The result of profiling of the runs on the medium suite for 32, 64, and 96 CPUs is also shown in Figure 4. The thread barrier probe is a function that spin waits until all threads have reached the barrier. The higher the percentage of time spent in this function, the higher is the load imbalance. Thus the load imbalance of the OMP medium version of WUPWISE increases with the number of processors. The percentage spent in the thread barrier probe in the OMP large version on 96 CPUs of WUPWISE is 18 %, while it is 27 % for the OMP medium version of WUPWISE. Thus the OMP large version of WUPWISE seems to have a better load balance than the OMP medium version.

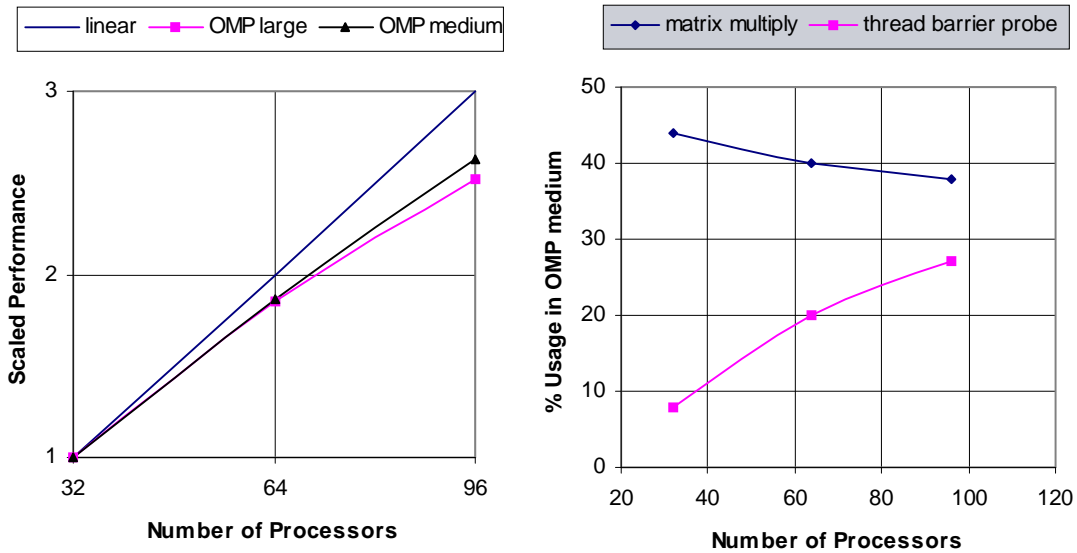


Figure 4: Scalability and Profiling of WUPWISE

## 5.2 MGRID

MGRID is a simple multigrid solver, which computes a 3 dimensional potential field. This code was parallelized using an automatic translator. first, and subsequently with manual improvements in order to avoid using multiple OMP PARALLEL / END PARALLEL constructs for consecutive parallel loops.

This benchmark was classified in [1] and [2] as having a good scaling to 64 processors. This classification is confirmed in this study, as can be seen in Figure 5. We see that MGRID in both the OMP medium and the OMP large scales well for 32 to 64 CPUs, but does not scale well from 64 to 96 CPUs. However the scalability is slightly improved in the OMP large benchmark suite.

The profiling results show that RESID does not scale from 64 to 96 CPUs, and that the relative weight of the thread barrier probe increases. The main costly routine is RESID which has an OMP PARALLEL DO with an iteration of a power of 2. The dominant calculation of RESID is that of 256 iterations in parallel. Therefore we need 128 CPUs to scale well from 64 CPUs, otherwise the resid must be parallelized by multiple dimensions.

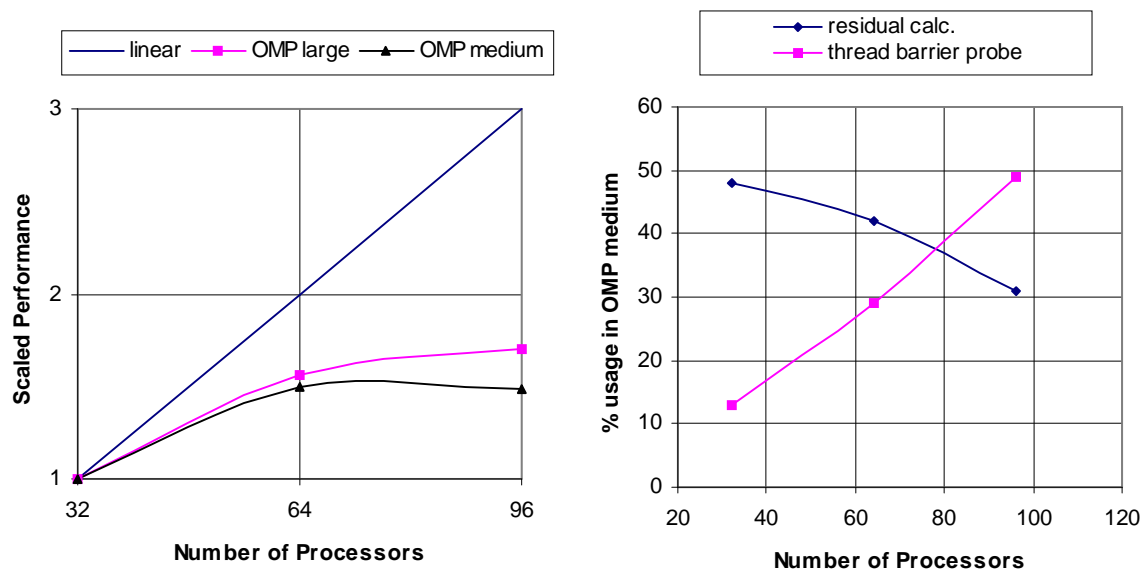


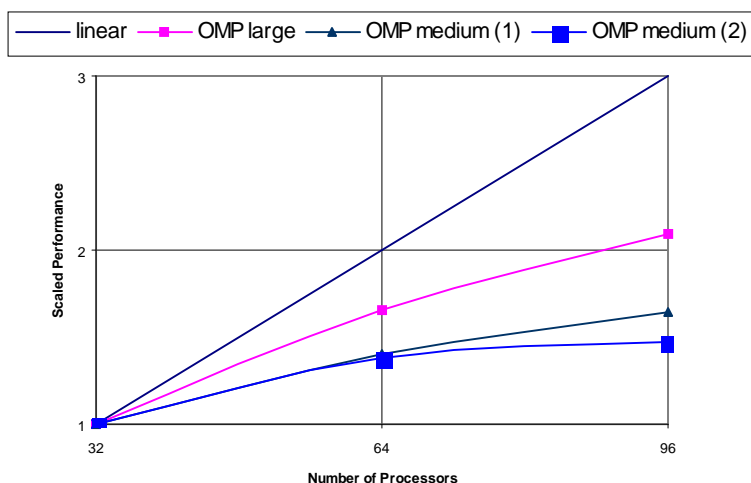
Figure 5: Scalability and Profiling of MGRID

### 5.3 FMA3D

FMA3D is a crash simulation code and the largest and most complex code in the benchmark suite. It simulates the inelastic, transient dynamic response of 3-dimensional solids and structures subjected to impulsively or suddenly applied loads. In [1] and [2], it was classified as a benchmark with poor scaling in the SPEC OMP medium suite. In this study we show that the scalability has a large improvement in the OMP large suite.

The difference between the medium size and the large size version of FMA3D is not only the size of the data. Many improvements of the parallelization of the code and reductions of the serial sections have been performed.

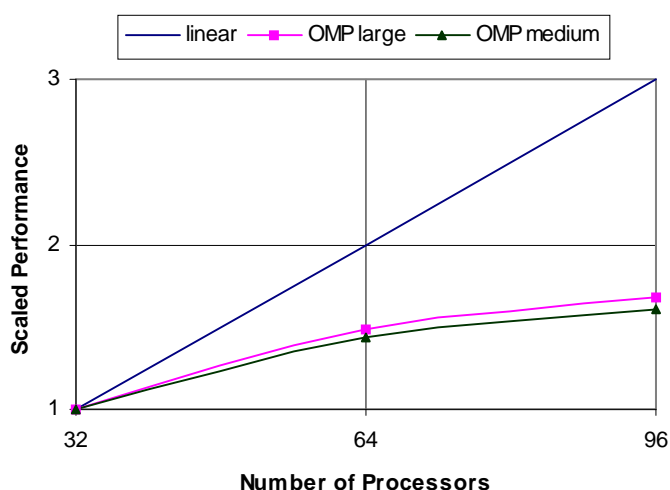
Figure 6 shows that the large size version of FMA3D shows good scalability up to 96 CPUs. The medium size version of FMA3D was classified as poorly scalable, but the large size version can be classified as a well scalable benchmark..



**Figure 6: Scalability of the FMA3D benchmark.**  
**(1) has been measured on the Primepower HPC2500 system, and (2) has been measured on the Primepower 2000 system.**

#### 5.4 SWIM

SWIM is a weather prediction model, which solves the shallow water equations using a finite difference method. It has large requirements with respect to the memory bandwidth. The scalability of the SWIM benchmark is shown in Figure 7. The difference between the large size and medium size version of this benchmark is not only in the size of the data. Several improvements of the parallelization of the code were performed and the number of serial sections in the code was reduced. The scalability between OMP large and OMP medium versions of SWIM is unchanged in spite of the source code change. The most costly routines in SWIM are calc3, calc2, and calc1. These routines have a good load balance but the computation time increases as the number of CPUs increase. We may be able to improve the scalability by rearranging the calculations and keeping the temporal locality.



**Figure 7: Scalability Results for SWIM**

## 6. Conclusion

We evaluated selected components of the Medium size and the Large size SPEC OMP2001 benchmark suites with the Parallelnavi Fortran and C/C++ compilers on the SMP server Fujitsu PRIMEPOWER HPC2500. We chose the components to be representative of three of the following categories : WUPWISE and SWIM that showed good scalability in OMP medium for all number of processors, MGRID that showed good scalability up to 64 processors, but poor scalability beyond 64 processors, and FMA3D that showed poor scalability.

We investigated the scalability of these components in the large dataset version of the benchmark suite, and compared the results with the scalability of the corresponding components of the medium dataset version of the benchmark suite. We have run the described components of the SPEC OMP benchmark on the soon to be released Fujitsu PRIMEPOWER HPC2500 system.

The main conclusions are (a) that FMA3D is well scalable in the OMPL large suite, whereas it was poorly scalable in the OMP medium suite, (b) that the scalability characteristics of MGRID have not changed, but that it requires a power of 2 in the number of processors to scale, (b) that the load balance in the OMP medium suite degrades with the number of CPUs, (c) that SWIM is scalable for both OMP medium and large suites, and the source code change in the OMPL large suite doesn't affect the scalability, and that (d) the OMP large suite has a better load balance than the OMP medium suite.

Without modification of benchmark codes (Base Metrics rule), the compilers have extracted a high and scalable performance for most benchmark codes in SPEC OMP2001.

We can also conclude that the PRIMEPOWER HPC2500 is a suitable platform for the implementation of OpenMP. With respect to the flat SMP hardware, it is shown that the execution performance increases smoothly with the number of CPU's without bending at any special numbers.

## 7. References

- [1] H. Saito, G. Gaertner, W. Jones, R. Eigenmann, H. Iwashita, R. Lieberman, M. van Waveren, and B. Whitney. Large System Performance of SPEC OMP2001 Benchmarks, In *Proc. Of WOMPEI2002, Lecture Notes in Computer Science*, Vol 2327, pp. 370-379, 2002.
- [2] H. Iwashita, E. Yamanaka, N. Sueyasu, M. van Waveren, and K. Miura. The SPEC OMP2001 Benchmark on the Fujitsu PRIMEPOWER System. In *Proc. of EWOMP2001*.
- [3] V. Aslot, M. Domeika, R. Eigenmann, G. Gaertner, W.B. Jones, and B. Parady. SPECComp: A New Benchmark Suite for Measuring Parallel Computer Performance. In *Proc. Of WOMPAT2001, Workshop on OpenMP Applications and Tools, Lecture Notes in Computer Science*, 2104, pages 1-10, July 2001. (<http://www.ece.purdue.edu/~eigenman/reports/wompat01spec.pdf>)
- [4] N. Izuta, T. Watabe, T. Shimizu, and T. Ichihashi. Overview of PRIMEPOWER 2000/1000/800 Hardware. *Fujitsu Sci. Tech. J.* 36(2):121-127, 2000. (<http://magazine.fujitsu.com/us/vol36-2/paper03.pdf>).
- [5] OpenMP Architecture Review Board. *OpenMP Fortran Application Program Interface Version 2.0*, November 2000. (<http://www.openmp.org/specs/mp-documents/fspec20.pdf>)
- [6] OpenMP Architecture Review Board. *OpenMP C/C++ Application Program Interface Version 1.0*, October 1998. (<http://www.openmp.org/specs/mp-documents/cspec10.pdf>)