John von Neumann Institute for Computing

Analyzing Mutual Influences of High Performance Computing Programs on SGI Altix 3700 and 4700 Systems with PARbench

Rick Janda, Matthias S. Müller, Wolfgang E. Nagel, Bernd Trenkler

published in

Parallel Computing: Architectures, Algorithms and Applications ,
C. Bischof, M. Bücker, P. Gibbon, G.R. Joubert, T. Lippert, B. Mohr, F. Peters (Eds.),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **38**, ISBN 978-3-9810843-4-4, pp. 373-380, 2007.
Reprinted in: Advances in Parallel Computing, Volume **15**,
ISSN 0927-5452, ISBN 978-1-58603-796-3 (IOS Press), 2008.

http://www.fz-juelich.de/nic-series/volume38

Analyzing Mutual Influences of High Performance Computing Programs on SGI Altix 3700 and 4700 Systems with PARbench

Rick Janda, Matthias S. Müller, Wolfgang E. Nagel, and Bernd Trenkler

Center of Information Services and High Performance Computing
Dresden University of Technology, 01162 Dresden, Germany
E-mail: rick.janda@zuehlke.com
E-mail: {matthias.mueller, wolfgang.nagel, bernd.trenkler}@tu-dresden.de

Nowadays, most high performance computing systems run in multiprogramming mode with several user programs simultaneously utilizing the available CPUs. Even though most current SMP systems are implemented as ccNUMA to reduce the bottleneck of main memory access, the user programs still compete in different ways for resources and influence the scheduler decisions with their generated load.

This paper presents the investigation results of the SGI Altix 3700Bx2 of the TU-Dresden and its successor system the Altix 4700 with the PARbench system. The PARbench system is a multiprogramming and multithreading benchmark system, which enables the user to assess the system behaviour under typical production work load and identify bottlenecks and scheduling problems.

The Altix 3700 and 4700 with their directory based ccNUMA architecture are the largest SMP systems on the market and promise a high scalability combined with a good isolation of the several system nodes. Several tests validates these promised features and analyzes the connection network and the utilized Intel Itanium 2 Madison (Altix 3700Bx2) and dual core Itanium 2 Montecito (Altix 4700) CPUs.

The paper will also show practical problems of the shared system bus by two CPUs each in the 3700 system and compare these situation with the internally shared system bus of the dual core CPUs in the 4700 system.

Further tests examine the load balancing behavior and it's consequences to OpenMP parallelized programs under overload.

1 Introduction

Nowadays, most high performance computing systems run in multiprogramming mode with several user programs simultaneously utilizing the available CPUs. Even though most current SMP systems are implemented as ccNUMA to reduce the bottleneck of main memory access, the user programs still compete in different ways for resources and influence the scheduler decisions with their generated load.

Section 2 gives a very brief overview of the architecture of the Altix systems and shows potential levels of influence. The PARbench system outlined in Section 3 is a powerful tool to analyze resource limitations and the mutual influence of programs. Section 4 analyzes the measured kernel data in order to give a first picture of the performance of the Itanium 2 CPUs and show the OpenMP scalability. The last section presents the load scenarios that analyzes the shared CPU bus, the general scalability and the load balance behaviour.

2 Memory Subsystem of the Altix 3700Bx2 and Altix 4700

The Altix 3700 and 4700 with the directory based ccNUMA architecture are the largest SMP systems on the market. Both systems mainly differ in the utilized Itanium 2 CPUs. An Altix 3700Bx2 system bundles each two Itanium 2 Madison CPUs together with one memory node together on a CPU module. Figure 1 illustrates the overall structure of such a CPU module. The two CPUs share a single system bus to the SHub. The architecture of the Altix 4700 follows the same schema but integrates one or two dual core Itanium 2 Montecito CPUs on the CPU module. The provided 4700 system only possesses one Montecito per module which makes the both systems especially comparable.

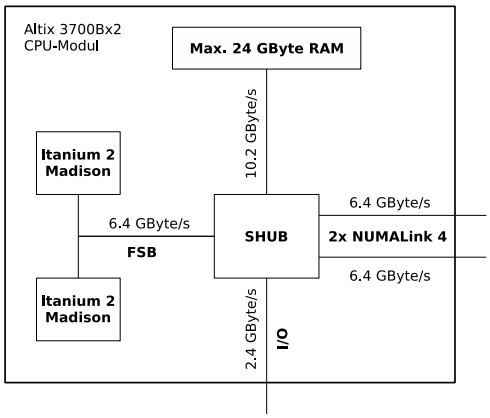

Figure 1. CPU board of an Altix 3700Bx2 system

Up to 256 CPU modules are connected by a NUMALink4 based dual plane fat tree and form a single ccNUMA system with a global shared main memory. The NUMALink connections ensure cache coherency over the whole system. The directory protocol integrated in each SHub reduces the amount of snooping signals for cache coherency by forwarding snooping signals to affected system nodes only. Together with an optimized memory placement that tries to allocate memory as near as possible to the CPU, the allocating process running on, the system architecture promises high scalability together with good isolation of the system nodes. Part of the evaluation was to validate these promised features.

3 Main Features of the PARbench System

OpenMP is a widely used approach to parallelize calculations on SMP systems. In the past, a lot of work has been presented that assesses the performance of OpenMP programs on dedicated SMP systems or measure the runtime performance under quite favourable circumstances and say nothing about the interaction of several user jobs in production environment.

PARbench was designed to address exactly this issue. The PARbench system enables the user to generate almost arbitrary load scenarios in benchmark capable runtime based

on synthetic benchmark kernels.

For this reason, 28 mathematical cores are combined with customizable data volumes, different strides in memory access and the OpenMP parallelization on customizable amounts of threads to a large amount of benchmark kernels with very different characteristics and very short runtime in the range of milliseconds. In addition the user can define different metrics based on hardware performance counters and measured times. Typical defined metrics could be Main Memory Transfer, FLOPS, Cache Transfers, Write Fraction, Bus-Snoops/s, I/O-Transfer, Load Balance or Parallelization Efficiency.

The initial Base Data Mode measures all defined metrics for every kernel on a dedicated system or subsystem and store them for later program generation. According to the described load scenarios, the PARbench system synthesizes sequences of the synthetic kernels, that create the described program behaviour and have the specified runtime on a dedicated system. The generation algorithm ensures uniform behaviour distribution across the whole runtime of each program and a good variation of the different metrics around the specified average behaviour. The subsequent execution of the whole load scenario measures the runtime and the CPU times of each program in the scenario. A GUI enables the user to analyze all measured results and compare them to each other.[7] contains a detailed description of the PARbench system.

4 Discussion of the Kernel Base Data

The evaluation of the measured kernel base data permits first conclusions about the CPU performance and the OpenMP scalability. For all following tests, the kernels were configured to the following parameters:

- Data volumes of 240 KByte, 2500 KByte and 25 MByte to place variations of each core in different levels of the memory hierarchy
- Predefined memory access strides (1, 2, 4, 8, 16, 32)
- OpenMP parallelization up to 32 threads (1, 2, 4, 6, 8, 12, 16, 24, 32)

The combination of these parameters with the mathematical cores generates an amount of about 3800 benchmark kernels.

4.1 Basic CPU Performance Data

Analyzing the measured metrics of the benchmark kernels outlines the general performance data of the utilized Itanium 2 CPUs. Figure 2 shows the achieved main memory transfer rate and the FLOPS of all sequential kernels on the Altix 4700 with the Montecito CPUs at 1.6 GHz. The older Madison in the Altix 3700Bx2 shows almost the same distribution but with less peak performance. For the Montecito, several kernels allmost reach the theoretical peak performance of 6.4 GFlops. The Madison could only reach about 5.2 GFlops at 1.5 GHz, which extrapolates to 5.5 GFlops at 1.6 GHz. The Montecito can leverage its improved cache subsystem here. A new second level instruction cache of one MByte was introduced in Montecito. Madison, in comparison, only possesses a 256 kByte shared second level cache for instructions and data. The conducted tests also showed, that

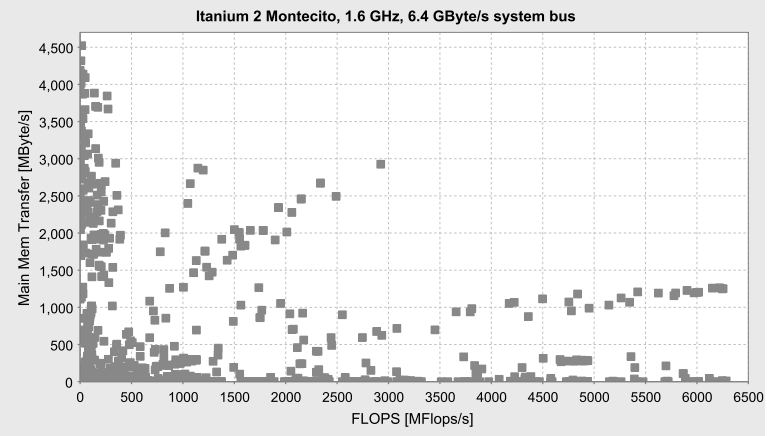

Figure 2. Floating point performance and main memory transfer of the utilized Itanium 2 Montecito (Intel Fortran Compiler 10.0.13 with highest optimization)

only the newest Intel compiler in version 10 was able to leverage the huge second level instruction cache, that was introduced in the Montecito. Compilations based on the version 9.1 reached only about 5.6 GFlops which corresponds to the extrapolated result from the Madison.

Considering the measured main memory transfer, it becomes clear that it is not possible to utilize the system bus with a single CPU respective a single CPU core. Madison and Montecito both only reaches about 4.5 GByte/s, which is far from the theoretical maximum of 6.4 GByte/s. This result is also supported by the work in[6].

4.2 OpenMP Scalability

In the given setup, the cores are parallelized with up to 32 OpenMP threads. Figure 3 shows the parallelization efficiency of all kernels with 32 threads. Many kernels perform very well with an efficiency of almost 100 percent. Some kernels show super linear speedup which is due to the increased overall amount of cache available with more CPUs. The kernels with very low efficiency are included by purpose and designed to stress the cache coherency protocol in multithreaded environments by means of accessing the data matrices in unfavourable direction. As of both systems use the same NUMAlink 4 based connection network, they both show equal results in this area.

5 Discussion of the Load Tests

5.1 Shared CPU bus as bottle neck

Each CPU board of the evaluated Altix 3700Bx2 system connects two Itanium 2 Madison CPUs over a shared system bus with the local SHub and therewith with the local memory node and the overall system. Other investigations have shown, that both CPUs together can utilize the available 6.4 GByte/s of the system bus in benchmark situations. The question

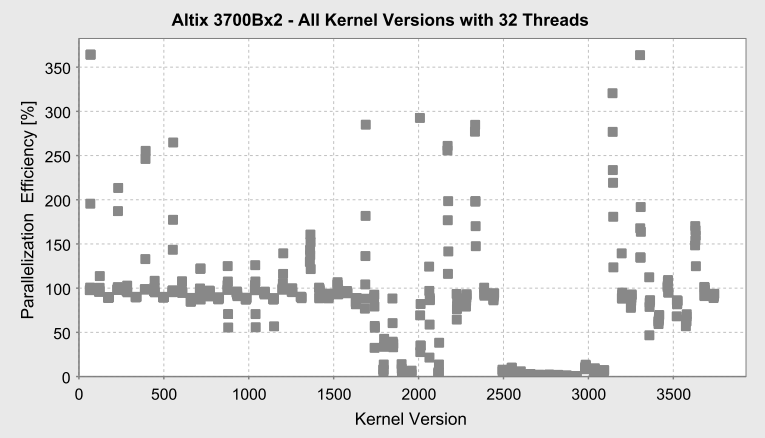

Figure 3. OpenMP scalability to 32 threads

is, how large is the impact in a practical environment, wherein both CPUs do not always use the bus in the same direction but create opposite requests. For this purpose, a load of eight sequential programs was executed on a subdivided CPU subset with four CPU boards. The programs were generated to an average main memory transfer of 2.14 GByte/s with mixed read/write access and 100 seconds runtime in a load free system. The memory transfer on the system bus of each of the four CPUboards should be about 4.4 GByte/s, which is far from the theoretical maximum of 6.4 GByte/s. Theoretically, the two programs should not interfere each other. The test result in Fig. 4 illustrates a different behaviour. The runtime of each program increases on average by 24.5 percent. The programs already influence each other considerably. The possible reasons could be bus read/write turn around cycles, that are necessary each time the transfer direction changes on the bus.

Figure 4. Runtime extension because of influences on the system bus (Altix 3700Bx2)

In order to determine the practical bus saturation, the test was repeated with different main memory transfer rates for all eight programs. The same tests were also conducted on the newer Altix4700 system, with one dual core Montecito CPU on each CPU board. Figure 5 shows the combined results of all these tests. Each bar represents the average runtime of all eight programs in a single load test. For the Altix 3700Bx2 system the measured performance impact impact is less than 10% only at a generated main memory transfer below 1 GByte/s per program. The two CPU cores in the Montecito show considerablely lesser mutual influence on their shared system bus which leads to only half the runtime extension in comparison to the two Madisons.

5.2 Overall System Scalability

To investigate the general system scalability, the first presented load test was repeated with larger amount of CPUs and appropriate amount of programs in the load scenario. The statistical examination does not show any changes in the runtime behaviour. The average runtime does not increase and also the min/max value remains the same.

5.3 Scheduling Behaviour Under Overload

Experiments with sequential overload clearly shows that the scheduling focuses on data locality and not on fair resource distribution. Figure 6 shows 17 sequential programs on 16 CPUs. 15 programs will be executed without interruption. The remaining two programs share one CPU until another CPU becomes available. According to the first touch policy, memory is allocated on the local memory node of a cpu board whereon a process or thread is executed at the time of allocation. Even if the process/thread is migrated to another CPU module, the data keeps on the same memory node and must be accessed by slower remote memory accesses. By binding processes and threads as long as possible to the same CPU, slower remote memory accesses are minimized. With this strategy, migrations are only

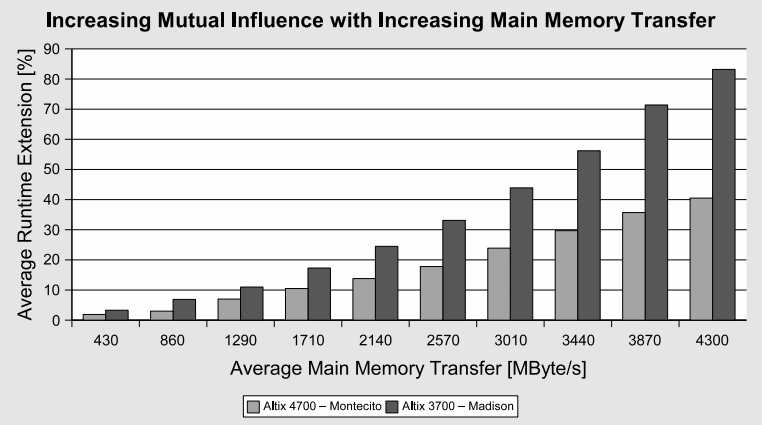

Figure 5. Runtime extension because of influences on the system bus

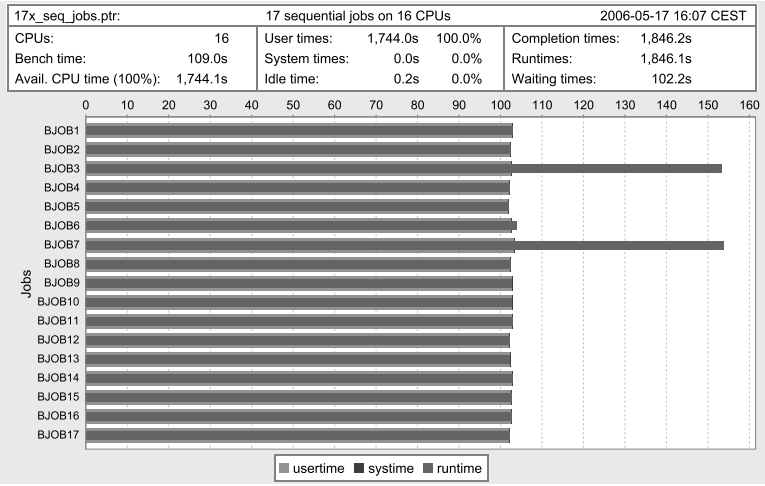

Figure 6. 17 sequential programs on 16 CPUs - focus on data locality

performed in order to utilize idle CPUs. Overload is not equally shared over all available CPUs.

This scheduling strategy has a strong influence on the behaviour of OpenMP programs and multithreaded programs in general. Figure 7 shows how multithreaded programs behave under overload. For this purpose, a scenario of 12 sequential jobs and four parallel jobs was executed on a 32 CPU subsystem.

Whereas the sequential programs in this high overload can almost finish their work undisturbed, the parallel programs substantially increase their runtime in comparison to the generated runtime in a dedicated environment. This behaviour does not significantly change, if the sequential programs are executed with lower scheduling priority. The reason for this stems from the scheduling behaviour and the way in which additional threads are spawned. If an additional thread is spawned by a process, the new thread is placed on the same CPU as the spawning process. The experiments with sequential overload already showed that the scheduler migrates only in order to utilize idle CPUs. Because all CPUs are already occupied by at least another process or thread, the additional threads of the OpenMP programs remain on the same CPU as the master thread. Under this circumstances, the further threads can not be leveraged and each OpenMP program is practically serialized on a single CPU.

6 Conclusion

The conducted tests showed the possible calculation power of the Itanium 2 CPUs. The improved cache subsystem of the Montecito leads to considerable more performance but only in conjunction with the newest Intel Fortran Compilers in version 10. The directory bases ccNUMA protocol together with the localized memory placement clearly confines the different system nodes and reduces the interaction effectively. High scalability and very good system node isolation could be clearly seen. The system bus of the CPUs showed

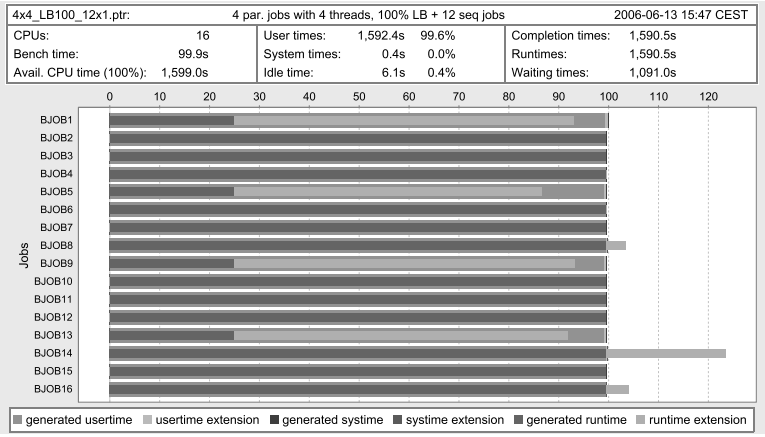

Figure 7. OpenMP programs under overload

some practical problems. One CPU/CPU core can not fully use it and two CPU/CPU cores influence each other to a large extent. With the integration of the two CPU cores on a single die in the Montecito the mutual influence over the system bus significantly decreased. The tests also showed that the scheduler clearly focuses on data locality and does not spread overload equally over the CPUs. Under these conditions, OpenMP programs practically becomes serialized under overload.

References

1. D. Lenoski et al., *The Standford Dash Multiprocessor* IEEE Computer Bd. **25**, 1992, 63–79, (1992).
2. M. Woodacre, D. Robb, D. Roe and K. Feind, *The SGI Altix 3000 Global Shared-Memory Architecture*, White Paper, (2003).
3. J. Aas, *Understanding the Linux 2.6.8.1 CPU Scheduler*, White Paper (2005). `http://josh.trancesoftware.com/linux/linux_cpu_scheduler.pdf`
4. Intel, Inc., *Intel Itanium 2 Processor Reference Manual for Software Development and Optimization*. `http://www.intel.com/design/itanium2/manuals/251110.htm`
5. SGI, *SGI NUMAlink - Industry Leading Interconnect Technology*, White Paper (2005). `http://www.sgi.com/pdfs/3771.pdf`
6. G. Juckeland, *Analyse der IA-64 Architektur: Leistungsbewertung und Möglichkeiten der Programmoptimierung*, Center for Information Services and High Performance Computing, Dresden University of Technology, Diploma thesis, (2005).
7. R. Janda, *SGI Altix: Auswertung des Laufzeitverhaltens mit Hilfe von neuen PARbench-Komponenten*, Center for Information Services and High Performance Computing, Dresden University of Technology, Diploma thesis, (2006).