**FORSCHUNGSZENTRUM JÜLICH GmbH**
**Zentralinstitut für Angewandte Mathematik**
**D-52425 Jülich, Tel. (02461) 61-6402**

Interner Bericht

# Comparison of DGEMM from ATLAS on one node of ZAMpano - sequential and parallel version, different releases -

*Inge Gutheil*

FZJ-ZAM-IB-2001-04

Juni 2001

(letzte Änderung: 12.06.2001)

# Comparison of DGEMM from ATLAS on one node of ZAMpano
# - sequential and parallel version, different releases -

Inge Gutheil

John von Neumann-Institut für Computing
Zentralinstitut für Angewandte Mathematik
Forschungszentrum Jülich GmbH

June 12, 2001

### Abstract

The first installed ATLAS BLAS library on ZAMpano, release 3.0 beta, only contained single processor BLAS for Pentium II processors. It showed good performance and could be used in combination with OpenMP. Now there is a version 3.2.1 for Pentium III multiprocessor machines which is already parallel. We compare the parallel version from ATLAS with the single processor ATLAS with OpenMP.

# 1 Introduction

The Basic Linear Algebra Subroutines BLAS [2] are the key to good performance of many numerical programs on today's computers. On many but not on all computers there are vendor optimized versions available. As an alternative there is the ATLAS (Automatically Tuned Linear Algebra Subroutines) [3] project which provides automatically tuned versions of the BLAS for a lot of architectures. ZAMpano [1] is a cluster of eight 4-processor Pentium III nodes with Linux operating system. The only BLAS on that computer were delivered with the Portland Group Fortran compiler, and they are not optimized. The performance of DGEMM there is up to 10 times slower than the performance of ATLAS BLAS.
The ATLAS BLAS, release 3.0 beta from December 1999, installed on ZAMpano in spring 2000, did not contain a version for multiprocessor Pentium IIIs. The only available architecture was a single processor version for Pentium II. As we wanted to study shared memory parallelism during a guest student program [5] we wrote an OpenMP [4] version of DGEMM, the routine for matrix-matrix multiplication. The matrix-matrix multiplication routine is the one with the highest performance as the computation to load ratio is best for this routine. So parallelization makes sense for this routine. OpenMP parallelization worked fine with the single processor ATLAS BLAS.
Now there is a new release, 3.2.1, available which contains a version for Pentium III 4 processor nodes like the ZAMpano nodes. There is a single-processor and a shared-memory parallel library. We compared the shared-memory parallel DGEMM from ATLAS 3.2.1, arch=PIII_4 with the OpenMP parallelized versions both with ATLAS 3.0 beta and 3.2.1 single processor. Additionally we compared the single processor versions from release 3.0 beta and 3.2.1. There were no significant differences.
We measured execution times and MFLOPS for DGEMM:

$$C = \alpha \text{op}(A) \times \text{op}(B) + \beta C$$

for square matrices $A,\ B$, and $C$ and all cases for op:

- NN = no matrix transposed
- NT = first matrix not transposed, second matrix transposed
- TN = first matrix transposed, second matrix not transposed
- TT = both matrices transposed

The times were measured with the routine rtc written in ZAM based on the run-time-clock. This was the only way to get high resolution timings. Additionally small problems were run 10-20 times and the execution time of running the routine several times was divided by the number of repetitions.
The MFLOPS were computed dividing the number of floating-point operations needed for a general matrix-matrix multiplication, $2n^3$ for square matrices of size $n$, by the execution times measured.

## 2 Results of Single-processor ATLAS DGEMM

Figures 1 and 2 show the same pattern for small matrices although the new release is slightly slower than the old one. Whereas the old version reaches about 400 MFLOPS for some matrix sizes smaller than $n = 108$ and the second matrix transposed the new version reaches only less than 390 MFLOPS. The difference gets smaller for matrix sizes between $n = 108$ and $n = 120$ and almost vanishes for $n = 128$ where the performance of both versions breaks in slightly in all cases.

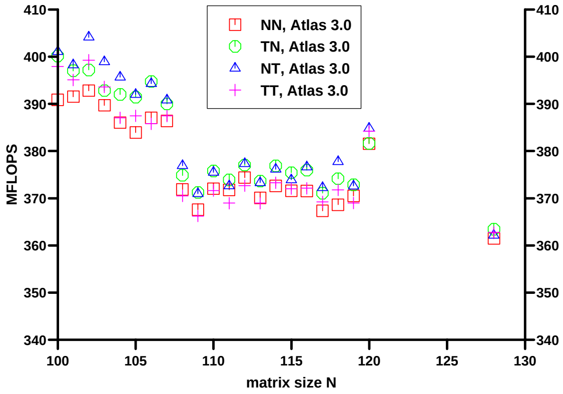

**Figure 1:** MFLOPS, small problems ($n = 100 - 130$), all cases, ATLAS release 3.0 beta

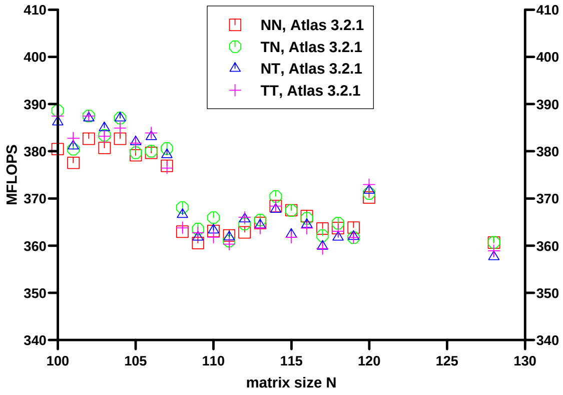

**Figure 2:** MFLOPS, small problems ($n = 100 - 130$), all cases, ATLAS release 3.2.1

On the other hand with the new release the differences between the four cases become smaller.

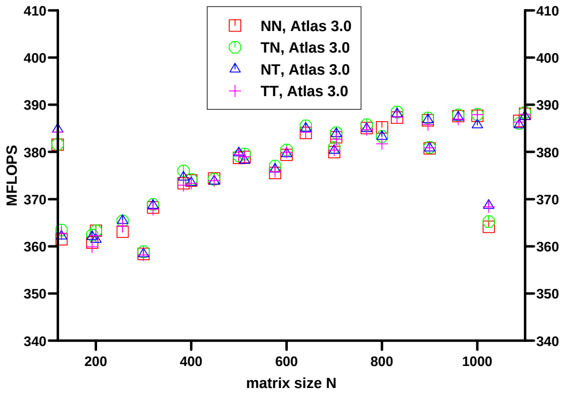

**Figure 3:** MFLOPS, medium size problems ($n = 120 - 1100$), all cases, ATLAS release 3.0 beta

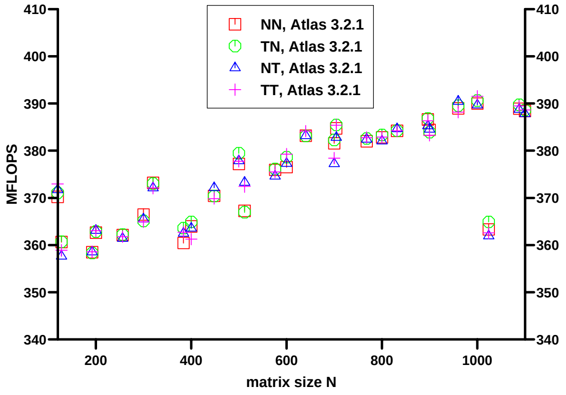

**Figure 4:** MFLOPS, medium size problems ($n = 120 - 1100$), all cases, ATLAS release 3.2.1

From figures 3 and 4 it can be seen that for matrix sizes between $n = 100$ and $n = 800$ both versions show comparable results and that for larger $n$ the new release becomes slightly faster.

Figures 5 and 6 both show significant performance degradations for $n = 1024$ and $n = 2048$, the latter is even stronger in the new release of ATLAS. The case with both matrices transposed becomes less efficient than the other cases for $n \geq 1800$ with ATLAS release 3.0 beta and for $n > 1400$ with ATLAS release 3.2.1. We don't have an explanation for this behaviour.

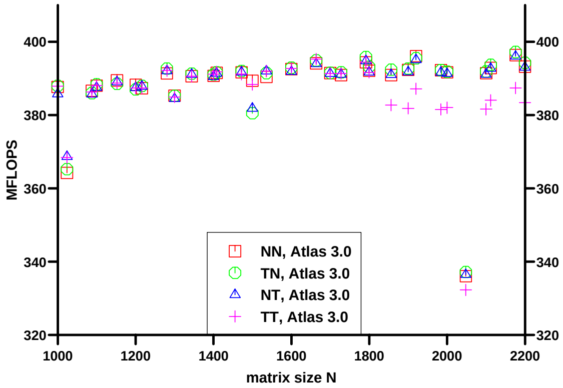

**Figure 5:** MFLOPS, large problems ($n = 1000 - 2200$), all cases, ATLAS release 3.0 beta

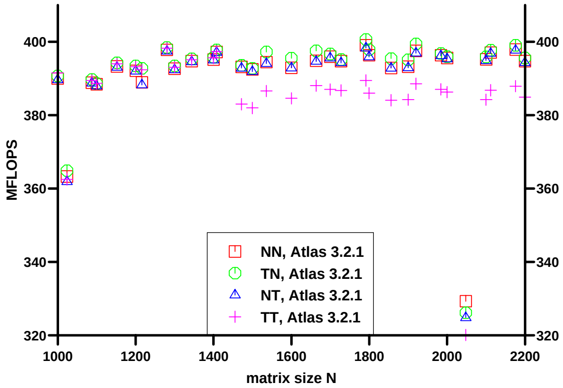

**Figure 6:** MFLOPS, large problems ($n = 1000 - 2200$), all cases, ATLAS release 3.2.1

## 3 Results of 4-processor ATLAS DGEMM

On the four processors of a single ZAMpano node we measured performance of a self-written DGEMMOMP using BLAS from ATLAS 3.0 beta and single node BLAS from ATLAS 3.2.1 and OpenMP as well as the 4-processor parallel DGEMM from ATLAS 3.2.1.

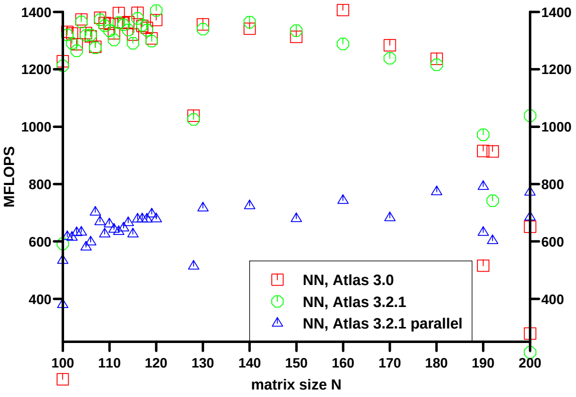

**Figure 7:** MFLOPS, no matrix transposed, ATLAS 3.0 and 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, only highest MFLOPS values ($n = 100 - 200$).

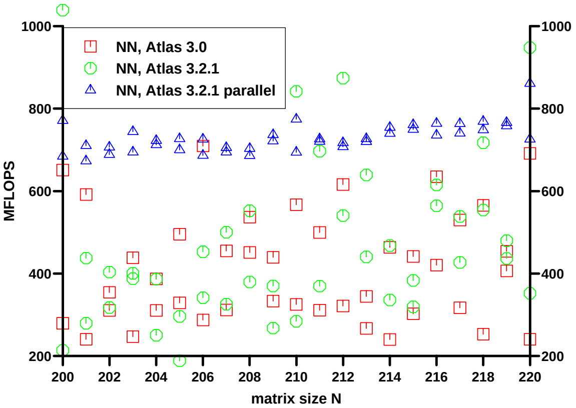

**Figure 8:** MFLOPS, no matrix transposed, ATLAS 3.0 and 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, highest and lowest MFLOPS values ($n = 200 - 220$).

We took the following matrix-sizes:$n = 64$ to 2048 in steps of 64, $n = 100$ to 2000 in steps of 100, $n = 100$ to 120 in steps of 1, $n = 1024$ to 4096 in steps of 64, $n = 4096$ to 7680 in steps of 256, $n = 100$ to 4000 in steps of 100, $n = 4200$ to 7500 in steps of 300, $n = 200$ to 220 in steps of 1, $n = 210$ to 1000 in steps of 30, $n = 100$ to 450 in steps of 10. The largest problem we could measure was $n = 7680$, for larger $n$ there was not enough memory to allocate all matrices and the test results.

For $n = 180,\ 450,\ 10$ we measured at least twice and took the fastest and the slowest execution time and MFLOPS for each $n$, as in that range those values often differed very much.

For $n = 100$ to $n = 200$ we often did only one measurement or we only took the fastest time, so the deviations cannot be seen in that range.

From figure 7 it can be seen that for very small problem sizes the MFLOPS values for parallel execution of the self-written routines can be rather high, whereas figure 8 shows that for $n > 200$ the parallel performance decreases strongly for the self-written routines. The MFLOPS values of the parallel ATLAS routines remain almost contant from $n = 100$ to $n = 220$.

The MFLOPS values of the self-written OpenMP version vary heavily whereas the parallel BLAS from ATLAS are rather stable for problem sizes between $n = 200$ and $n = 220$ and they are in general faster than the self-written ones.

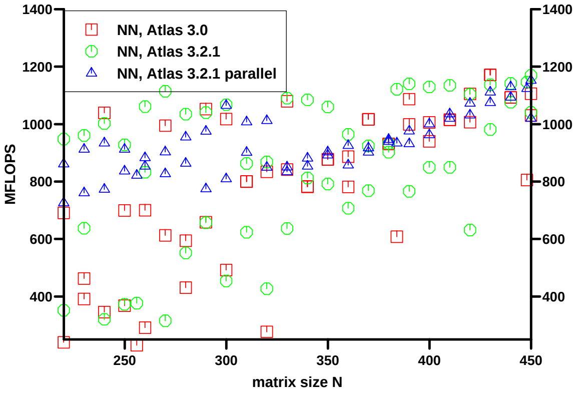

**Figure 9:** MFLOPS, no matrix transposed, ATLAS 3.0 and 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, highest and lowest MFLOPS values ($n = 220 - 450$).

Figure 9 shows that for $n > 220$ the self-written OpenMP routines can be faster and slower than the ATLAS parallel ones as the highest MFLOPS value is often higher and the lowest lower than the one of the ATLAS parallel routine. The performance of all versions converge as the matrix size approaches $n = 450$, so for $n > 450$ we only measured once or took the highest performance.

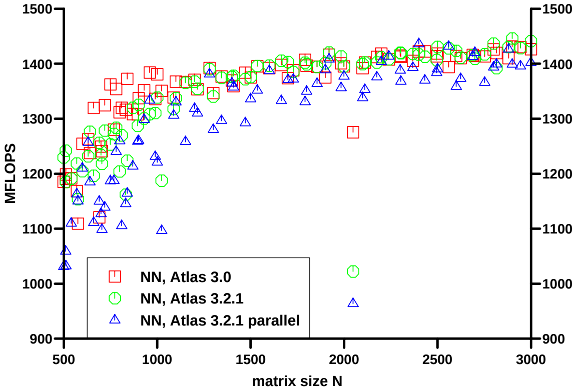

**Figure 10:** MFLOPS, no matrix transposed, ATLAS 3.0 and 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, only highest MFLOPS values ($n = 500 - 3000$).

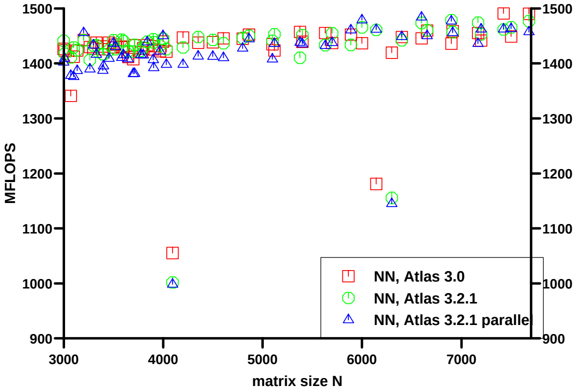

**Figure 11:** MFLOPS, no matrix transposed, ATLAS 3.0 and 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, only highest MFLOPS values ($n = 3000 - 7700$).

For $n$ being a power of two, i.e. $n = 1024,\ 2048$, and $4096$ and also for $n = 6144$ there is a ”break down” of the performance with all versions. The performance of the ATLAS parallel routine is generally slower than the performance of the self-written OpenMP routine for $n < 4500$ and is higher or the same for larger $n$.

For small matrices the results are almost identical for the three cases NN, NT, and TT. The performance for small matrices changes dramatically in the case TN. Figures 12 to 14 show that the performance of the self-written OpenMP code is almost constant and nearly as high as for large matrices. The ATLAS parallel code performs significantly slower for matrix sizes up to 200 and still slower for matrix sizes up to 400.

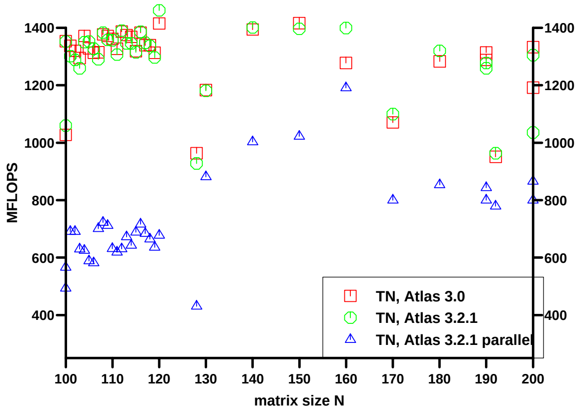

**Figure 12:** MFLOPS, first matrix transposed, ATLAS 3.0 and 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, only highest MFLOPS value ($n = 100 - 200$).

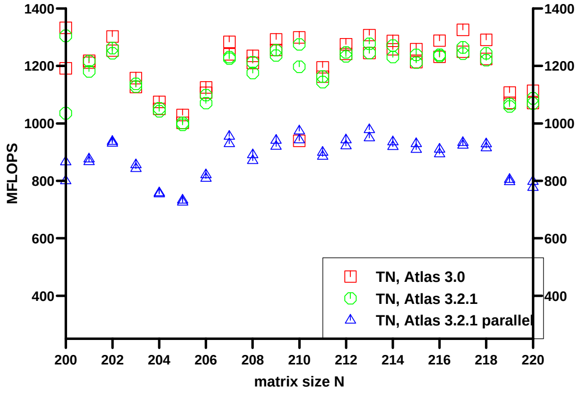

**Figure 13:** MFLOPS, first matrix transposed, ATLAS 3.0 and 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, highest and lowest MFLOPS value ($n = 200 - 220$).

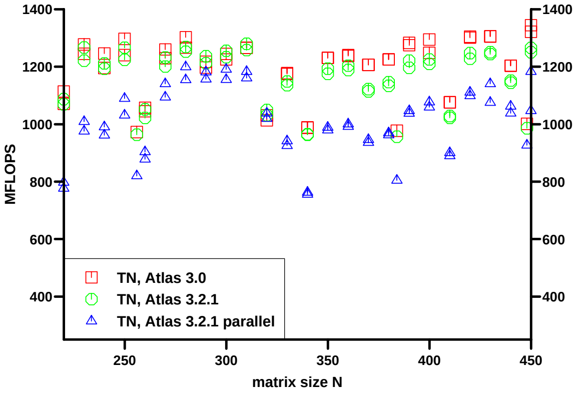

**Figure 14:** MFLOPS, first matrix transposed, ATLAS 3.0 and 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, highest and lowest MFLOPS value ($n = 220 - 450$).

For $n \geq 500$ the results are similar in all cases. For $n > 4000$ there is no significant difference in performance between the different cases and between self-written OpenMP code and ATLAS parallelized version.
For very small matrices, i.e. $n < 200$ and for matrices with $500 \leq n \leq 4000$ the self-written OpenMP parallelized routine DGEMMOMP is often faster than the ATLAS parallelized one, for the rest of the matrix sizes it is the other way round. So in general it is not necessary to parallelize on ones own, the parallel ATLAS BLAS are rather good.

# 4 Speedups

Speedups were only measured for ATLAS BLAS release 3.2.1 as single processor BLAS and for matrix sizes up to $n = 4200$ (as the execution times on one processor exceeded 5 minutes for $n > 3900$). We measured speedup by dividing the execution time with the single processor ATLAS 3.2.1 by the execution times with the self-written OpenMP version with ATLAS 3.2.1 single processor Blas and the parallel version of ATLAS 3.2.1.

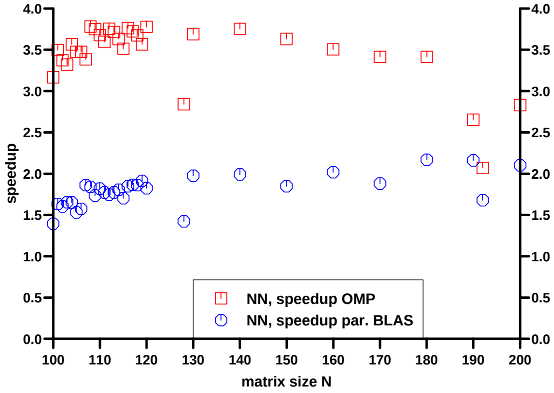

**Figure 15:** Speedups, no matrix transposed, ATLAS 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, only shortest execution times ($n = 100 - 200$).

Now we always took only the shortest execution times even in those cases where shortest and highest execution times differed very much.

For the three cases, no matrix transposed, second matrix transposed, and both matrices transposed, the speedup curves are almost the same which is not surprising as the MFLOPs rates were almost the same, too.

Figures 15 and 16 show excellent speedup values of often more than 3.5 for very small problem sizes of $n \leq 180$ with the self-written OpenMP parallelized routine. The speedup then decreases dramatically to often less than 2 and even a slow-down can be seen in some cases even for the fastest execution of the OpenMP parallel routine. For $n > 250$ the speedup starts exceeding 2.5 with some ”break-ins” until it stabilizes at a value a little larger than 3 for $400 < n < 500$.

The speedup of the parallel BLAS routine from ATLAS 3.2.1 on the other hand is rather stable for all values on $n$. It starts at 1.5 for $n < 120$ and remains about 2 until $n = 219$, then it slightly increases (again with some ”break-ins”) to values about 3.5-3.6 for $n > 2000$.

Figure 16 shows that at least for $200 \leq n \leq 220$ the speedup of the parallel routine from ATLAS 3.2.1 is higher than the speedup of the self-written OpenMP routine,

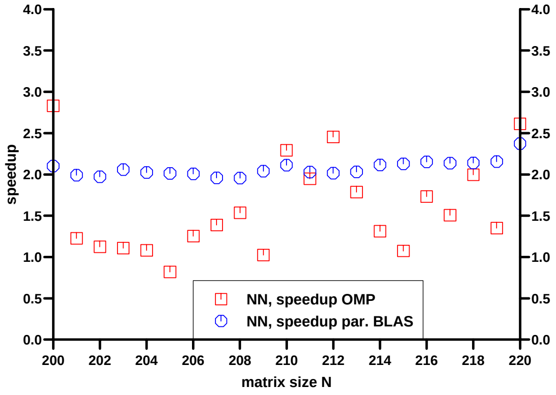

**Figure 16:** Speedups, no matrix transposed, ATLAS 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, only shortest execution times ($n = 200 - 220$).

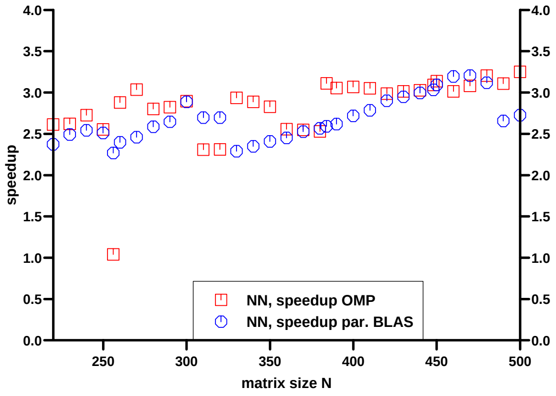

**Figure 17:** Speedups, no matrix transposed, ATLAS 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, only shortest execution times ($n = 220 - 500$).

but figures 18 and 19 show that for $n > 500$ more often the self-written routine reaches the higher speedups.

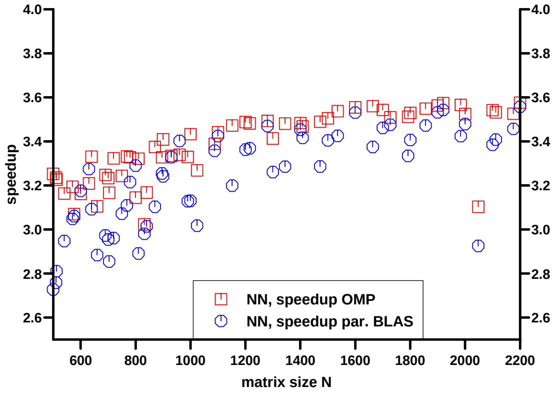

**Figure 18:** Speedups, no matrix transposed, ATLAS 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, only shortest execution times ($n = 500 - 2200$).

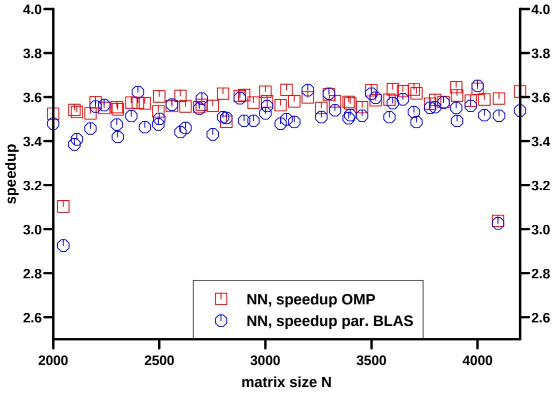

**Figure 19:** Speedups, no matrix transposed, ATLAS 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, only shortest execution times ($n = 2000 - 4200$).

All speedup curves show that a parallelization of BLAS routines on one node of ZAMpano makes sense for matrix sizes $n > 300$ and for $100 \leq n \leq 120$ with OpenMP using ATLAS single processor BLAS.
Using the parallel BLAS from ATLAS 3.2.1 pays off for matrix sizes of $n > 500$ and although it is often slightly slower than self-written OpenMP parallelized BLAS, there is no real need for the self-written routine.

As expected from the MFLOPS the case with the first matrix transposed shows different speedup behaviour than the other three cases for $n < 500$ and almost the same behaviour for larger $n$.
The speedup for small problems remains at a value of about 3.5 with the self-written OpenMP parallelized BLAS and about 2.5 for $n \leq 220$ and 3.0 for $220 < n \leq 500$ with the parallel ATLAS BLAS. With growing $n$ it approaches a value of 3.6 as in all other cases.

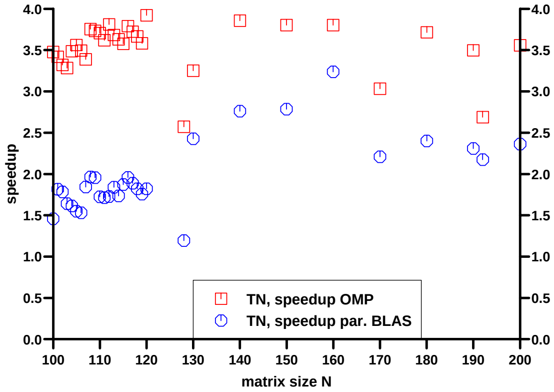

**Figure 20:** Speedups, first matrix transposed, ATLAS 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, only shortest execution times ($n = 100 - 200$).

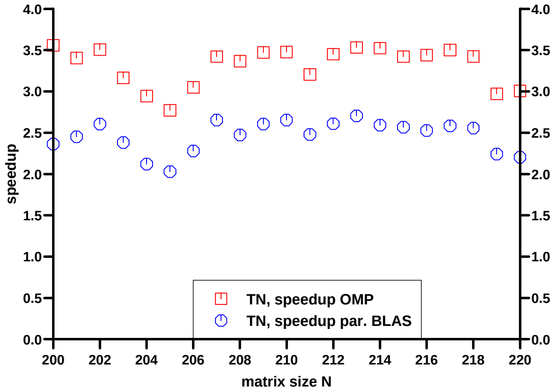

**Figure 21:** Speedups, first matrix transposed, ATLAS 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, only shortest execution times($n = 200 - 220$).

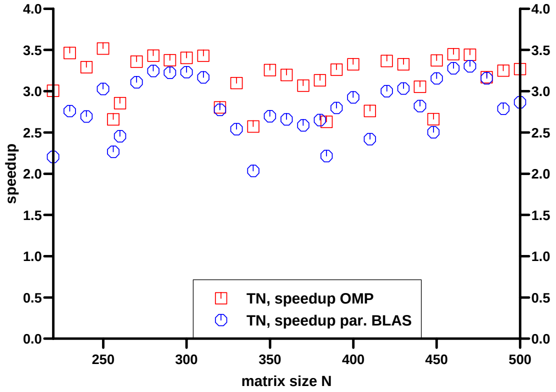

**Figure 22:** Speedups, first matrix transposed, ATLAS 3.2.1 with OpenMP, ATLAS 3.2.1 parallel version, only shortest execution times ($n = 220 - 500$).

# References

[1] *ZAMpano - ZAM Parallel Nodes*
http://zampano.zam.kfa-juelich.de

[2] *BLAS - Basic Linear Algebra Subprograms*
http://www.netlib.org/blas

[3] *ATLAS - Automatically Tuned Linear Algebra Software*
http://www.netlib.org/atlas

[4] *OpenMP*
http://www.openmp.org

[5] T. Betcke, *Performance Analysis of various parallelization methods for BLAS3 routines on cluster architectures*, in: Beiträge zum Wissenschaftlichen Rechnen, Ergebnisse des Gaststudentenprogramms 2000 des John von Neumann-Instituts für Computing, R. Esser, D. Mallmann (Hrsg.), Internal Report, FZJ-ZAM-IB-2000-15