

# The performance of NAMD on HPCx

Joachim Hein  
HPCx Terascaling Team  
EPCC  
The University of Edinburgh

December 15, 2003

## Abstract

We report on the performance of the NAMD package on the HPCx system using up to 1024 processors. This study includes a comparison of the most recent versions 2.4 and 2.5. We observe a substantial performance improvement for version 2.5 over 2.4. For larger configurations, we observe good scaling to 256 or more processors. NAMD proves to be sensitive to `MP_EAGER_LIMIT` and users have to tune this environment variable to achieve good performance.

## 1 Introduction

Currently massively parallel supercomputers are capable of simulating biomolecular systems containing several 100000 atoms. In this report, we investigate the performance of the NAMD package on the HPCx system. NAMD is a molecular dynamics code designed for the simulation of large biomolecular systems. One of NAMD's key aims is the efficient use of modern massively parallel computer architectures [1].

We start this report with a review of the HPCx system. For the performance investigation in section 3 we used four different input files. These files describe systems ranging from 23558 atoms to more than 327000 atoms. We report on the time per simulation step and the parallel efficiency achieved on the system. We show that users have to adjust the environment variable `MP_EAGER_LIMIT` to achieve good performance. The load balance and floating point performance is a another key point of this investigation, see section 3.4. Section 4 contains a discussion of our results.

## 2 HPCx system

### 2.1 Hardware

HPCx consists of 40 IBM p690 Regatta H frames. Each frame has 32 POWER4 processors with a clock speed of 1.3 GHz. This provides a peak performance of 6.6 Tflop/s and up to 3.2 Tflop/s sustained performance. The frames are connected via a dual plane IBM SP Switch2 (Colony) network. Per frame, the processors are grouped into 4 *multi chip modules* (MCM). Each MCM has 8 processors. In order to increase the communication band width of the system, the frames have been divided into 4 *logical partitions* (LPAR), coinciding with the MCMs. Each LPAR is operated as an 8-way SMP, running its own copy of the operating system AIX. Users are granted exclusive access to the LPARs allocated. They are charged per LPAR use, irrespective of the number of processors per LPAR they use.

The processors inside an LPAR share the same memory architecture, in particular there is only a single bus to main memory and also a single level 3 cache of 128 MB per LPAR or MCM. The level 2 cache of 1440 kB is shared between 2 processors and each processor has its own level 1 cache of 32 kB.

## 2.2 Run-time environment

Unless stated otherwise, we set the environment variable `MP_EAGER_LIMIT=65536`.

# 3 Performance

## 3.1 Input files

We used four different input files for this performance investigation, describing biomolecular systems between 23558 and 327506 atoms.

1. The smallest system under investigation has **23558** atoms and describes dihydrofolate reductase in a water bath. This system is known as the “*Joint Amber-Charm*” Benchmark (JAC) and is available from Charles L. Brooks III, Scripps Research Institute [2].
2. The second system is the well known Apo A-1 benchmark provided by the NAMD developers [3]. This file contains **92224** atoms.
3. The next system describes the interactions between TCRs and peptide-MHC [4]. This system contains **96796** atoms, which is of a similar size to Apo A-1. The file was made available to us for this investigation by Peter Coveney, University College London.
4. The largest system uses **327506** atoms and describes the  $F_1$  subunit of ATP synthase [5]. These files were made available to us by Jim Phillips, University of Illinois at Urbana-Champaign and Rick Kufrin, NCSA.

## 3.2 Runtime and parallel efficiency

NAMD optimises its performance during execution. This is obvious from its output files. In particular when using a large number of processors, the first 200 steps take substantially longer than the following steps. During the initial steps the program measures the time needed to calculate various objects and redistributes these objects onto the processors to improve the load balance. For this study we typically performed about 1000 steps to measure the performance, which takes a few seconds to complete when using a large number of processors.

A typical production sized run of 500000 steps takes several hours to complete [4]. For these runs the time taken for the slower initial steps is normally negligible, as is the time needed to read the input files<sup>1</sup>. Hence to obtain reliable estimates for the cost of production runs from short and inexpensive test runs, we have excluded the initial slow iterations and the set up costs for the simulation. For this purpose NAMD reports a “*Benchmark time*” at various stages of its execution. The benchmark is an average over several iterations after the initial load balancing steps. For several runs, we compared the “*Benchmark time*” to the average time for steps 301 to 1000. In practice, these times are equal.

For the four different test configurations we will report on the time per step multiplied by the number of processors. For a code which exhibits good scaling, plotting this value against the number of processors results in an almost horizontal line.

In Figure 1 we show the performance of NAMD 2.4 and 2.5 for the dhf reductase. The performance was measured for up to 128 processors. The first thing to note, for a given number of processors, the performance of NAMD version 2.5 is superior to version 2.4. For both versions, NAMD exhibits

---

<sup>1</sup>For illustration: For the TCR peptide-MHC system reading the input files takes about three minutes.

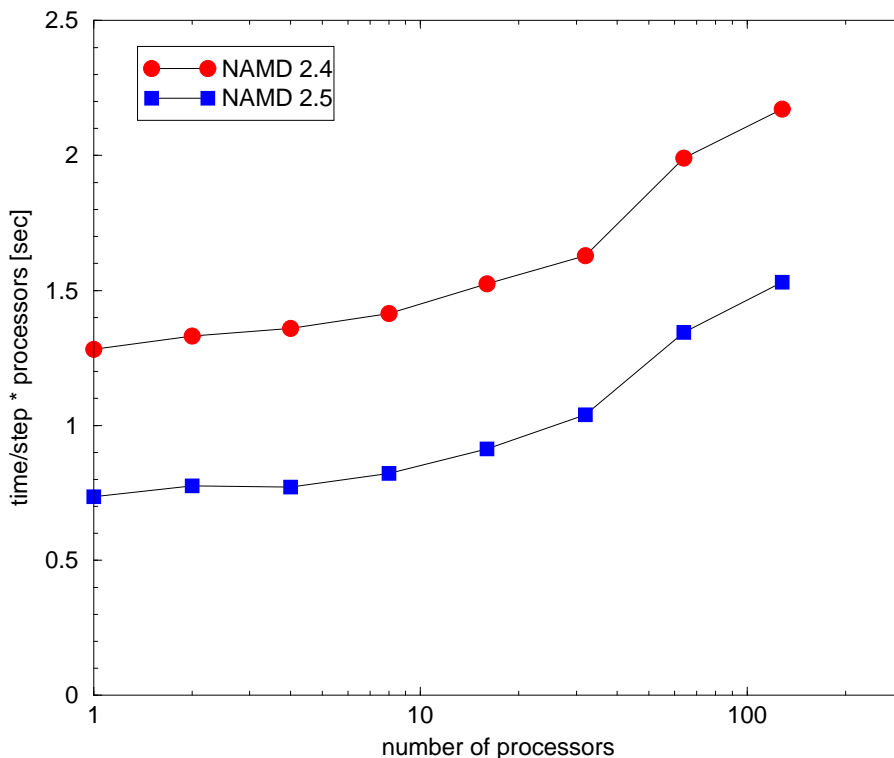


Figure 1: Performance comparison between NAMD version 2.4 and 2.5 using the dhf reductase system with 23558 atoms[2]

good scaling up to 32 processors. Above 32 processors the execution time still decreases, however the scaling is reduced.

In Figure 2 we show the results for the Apo A-1 benchmark. Again NAMD 2.5 shows vastly superior performance, when compared to version 2.4. For both versions, the code shows good scaling for up to 256 processors. When using more than 256 processors, the calculation becomes increasingly expensive, though the time still decreases when using up to 1024 processors.

For the Apo A-1 benchmark we also studied the dependency on the number of active processors per LPAR. The results of this study are included in Figure 2. The performance of NAMD shows very little dependence on the number of processors per LPAR. This indicates that the code does not use the bus from Level 3 cache and main memory to its limit. See [6] for an example for benchmark which strongly depends on the number of active processors per LPAR. As a result of this part of the investigation we recommend HPCx users of NAMD to use 8 active tasks per LPAR.

Figure 3 shows NAMD's performance when using the TCR peptide-MHC system. The observed performance is approximately equal to the Apo A-1 system, which contains a similar number of atoms.

The performance of the large  $F_1$  ATP synthase system containing 327506 atoms has been investigated for NAMD 2.5 and is shown in Figure 4. For this system the increase in cost when using 1024 processors is quite moderate when compared to the other systems under investigation, which is shown more clearly in Figure 5.

Figure 5 compares the parallel efficiencies for the four different biochemical systems under investigation. We have chosen 8 processors from a single LPAR as the point of reference. We define the parallel efficiency  $E_8(N)$  with reference to eight processors as

$$E_8(N) = \frac{8t(8)}{Nt(N)}.$$

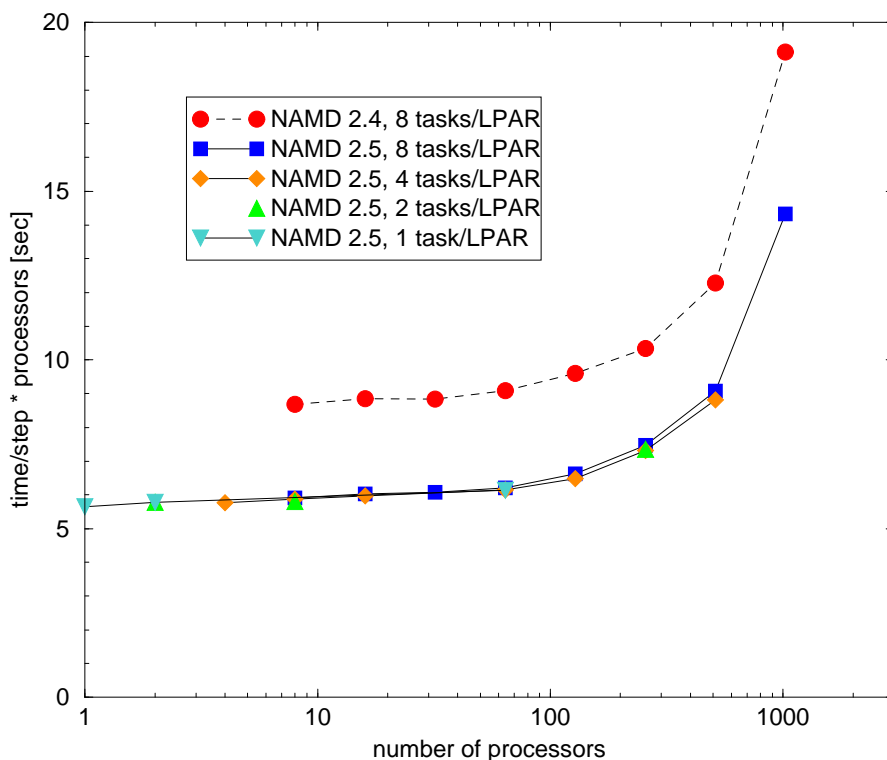


Figure 2: Performance comparison between NAMD version 2.4 and 2.5 using the Apo A-1 benchmark [3]. This system simulates 92224 atoms.

Here  $t(N)$  denotes the time per step when using  $N$  processors. On HPCx eight processors is a natural choice of reference, since users are charged with respect to LPARs used and we have discovered that it is most efficient to choose eight processors per LPAR.

As expected, the figure shows the code scales better for a larger system. The performance for Apo A-1 and TCR peptide-MHC is about equal, which is not surprising since these systems have comparable atom counts.

### 3.3 Dependence on `MP_EAGER_LIMIT`

The environment variable `MP_EAGER_LIMIT` controls the protocol used for sending messages between the processors. For messages of a size below its value, a buffered send is used, leading to a lower latency. For message of a size larger than `MP_EAGER_LIMIT` a synchronous send is used. Using a synchronous send reduces the need for buffer space and increases the number of outstanding messages the system can cope with. The system defaults are very small when using a larger number of processors.

The dependency of the performance of NAMD on the value `MP_EAGER_LIMIT` was investigated for the TCR peptide-MHC system on 256 processors. For this number of processors, the default value of `MP_EAGER_LIMIT` is 256 bytes. The results are shown in Figure 6. For both versions of NAMD the performance improves dramatically when increasing the value of `MP_EAGER_LIMIT`. The system default results in poor performance and users are encouraged to change the value of `MP_EAGER_LIMIT` in their load-leveler scripts. When using ksh include the line:

```
export MP_EAGER_LIMIT=65536
```

into the load-leveler scripts before calling NAMD.

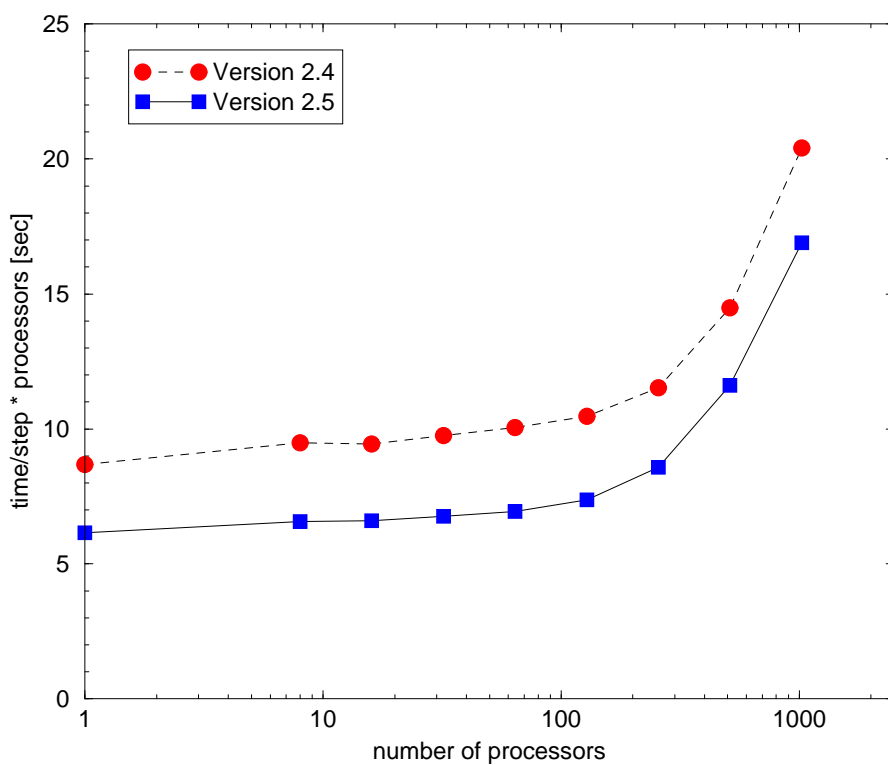


Figure 3: Performance comparison between NAMD version 2.4 and 2.5 using the TCR peptide-MHC system [4]

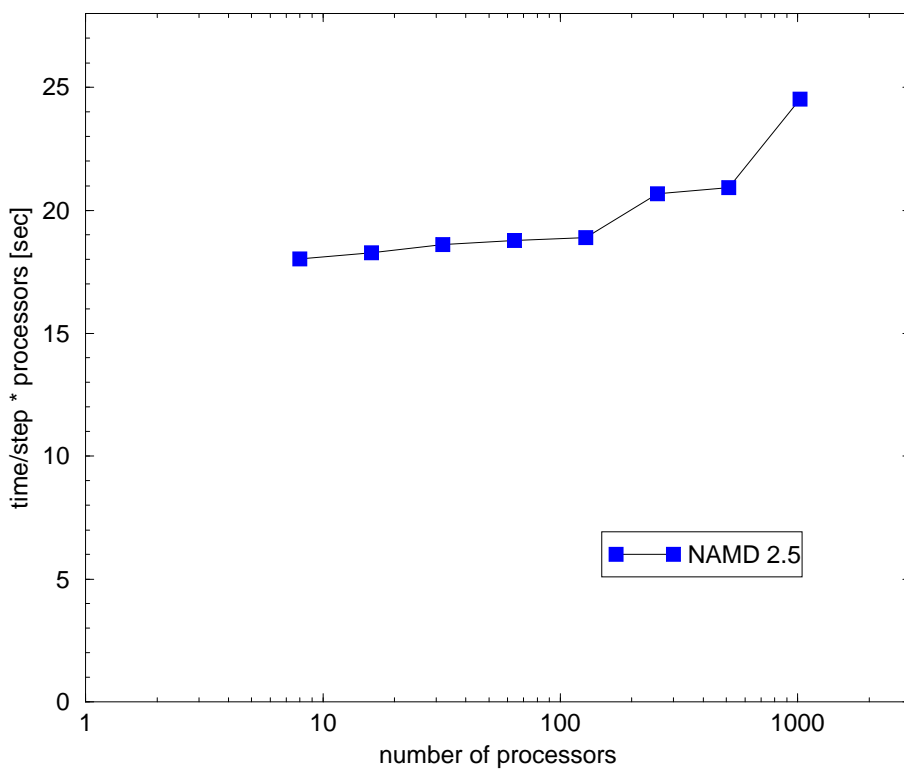


Figure 4: Performance and scaling of NAMD 2.5 for the large 327506 atom ATPase system [5]

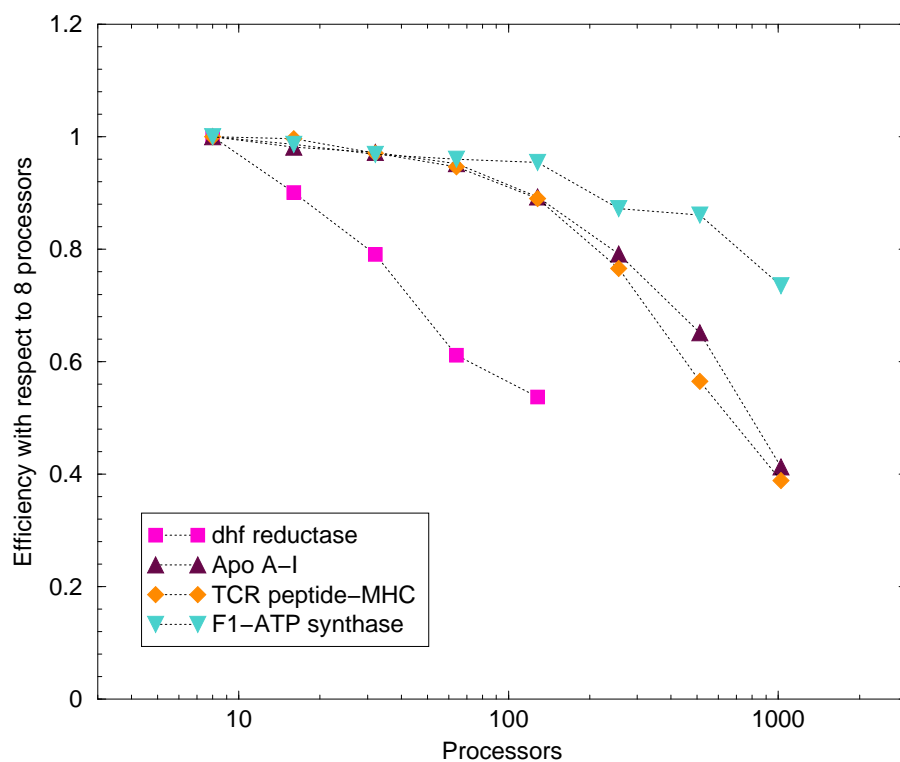


Figure 5: Comparison of the parallel efficiencies for the four benchmarks under investigation when using NAMD 2.5. The efficiency has been calculated with respect to a single fully populated LPAR, which corresponds to eight processors.

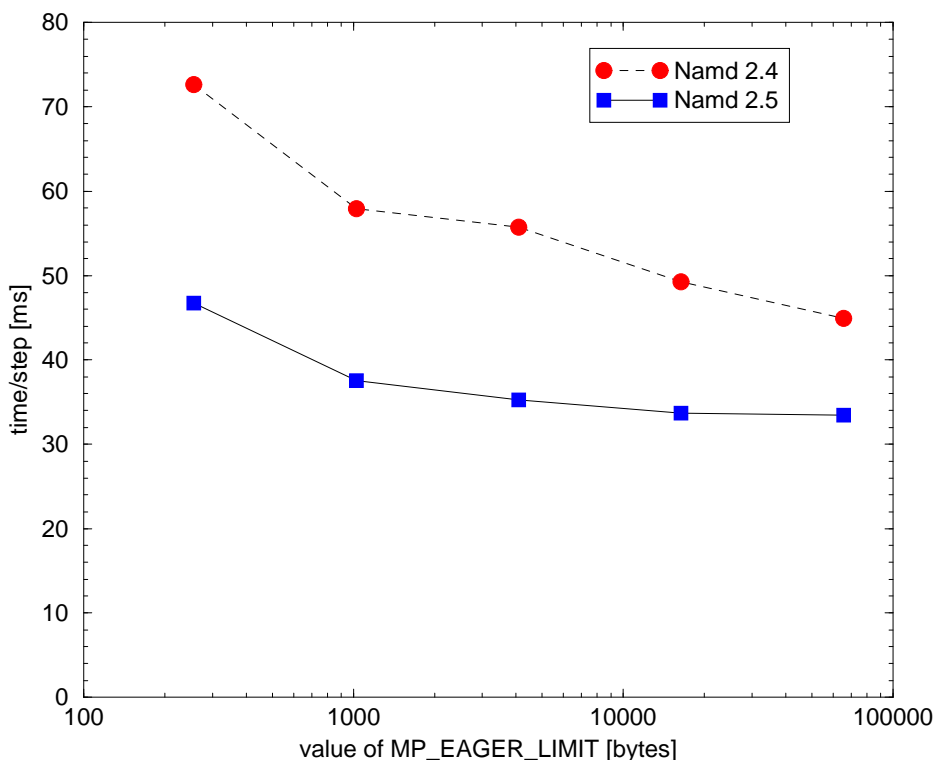


Figure 6: Effect of the performance on the environment variable `MP_EAGER_LIMIT`. Results are for the TCR peptide-MHC system [4] with 96796 atoms on 256 processors.

### 3.4 Floating point performance and load balance

In this section we investigate the floating point performance and the load balance of the application. This was investigated for 8 and 128 processors using `hpmcount` [7, 8]. Since `hpmcount` measures the performance across the entire application, we increased the number of step to 2000 for the 8 processor runs and 32000 for the 128 processor runs. This was done to minimise the effect of the overheads, such as program start up, reading the input files and the initial load balancing steps.

In Figure 7 we show histograms of the distribution of the flop rates across the processors. For both versions of NAMD, the histograms show that the code is very well balanced. When using 128 processors, there is always one processor which has a substantially lower workload than the others. However for 128 processors the impact of this single processor has a negligible effect on the overall performance.

It is noticeable that the older version has a significantly higher flop rate than the present version 2.5. This indicates that there are algorithmic changes between NAMD 2.4 and 2.5, which as we have show in the previous section, lead to a faster time to solution with fewer floating point operations.

The histograms shows that the flop rate reduces with larger numbers of processors. However for the NAMD 2.5 this reduction is smaller than that observed for NAMD 2.4.

## 4 Discussion

We have investigated the performance of NAMD on the HPCx system. Using a variety of biomolecular systems we observe very good scalability and for large systems NAMD can make efficient use of the 1024 processors available on HPCx.

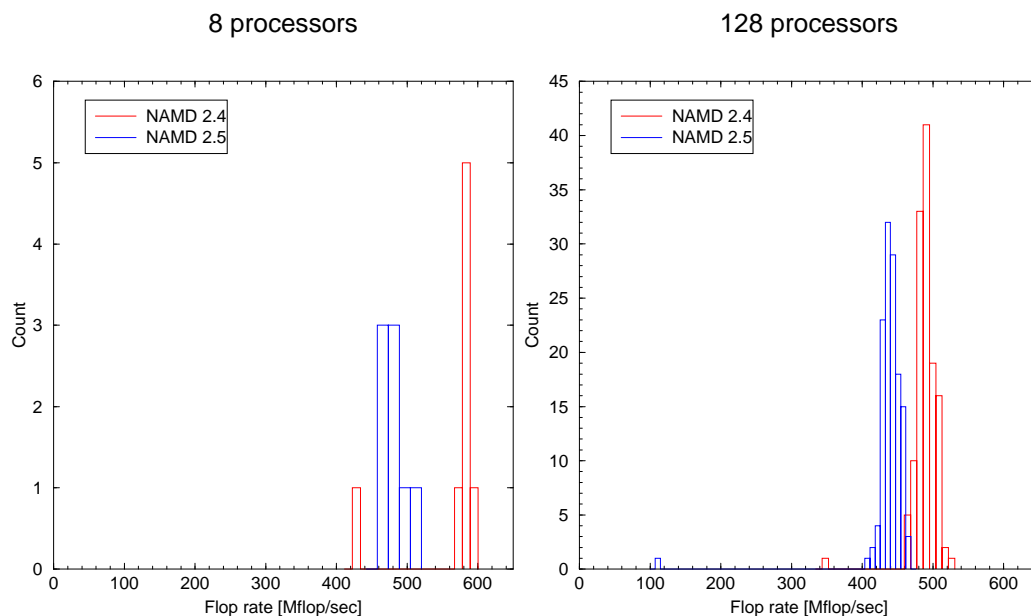


Figure 7: Distribution of the flop rates on the processors for 8 and 128 processors. Results are for the 96796 atom TCR peptide-MHC system [4].

We observe a substantial performance improvement for version 2.5 over the previous version 2.4. Our investigation show that tuning the environment variable `MP_EAGER_LIMIT` is essential to achieve good performance on HPCx. The distribution of the flop rates onto the processors indicates that the dynamic load balancing techniques employed by the application are effective.

## Acknowledgement

I would like to acknowledge useful discussions with Mark Bull, Peter Coveney, Rick Kufrin, Jim Phillips, Lorna Smith and Kevin Stratford. In particular, I would like to thank Peter Coveney, Rick Kufrin and Jim Phillips for making their configuration files available to us.

We would like to acknowledge EPSRC's financial support for the HPCx service and this investigation.

## References

- [1] L. Kalé, et al., *NAMD2: Greater scalability for parallel molecular dynamics*. *Journal of Computational Physics*, 151:283-312, 1999.
- [2] Joint Amber Charm (JAC) benchmark, [www.scripps.edu/brooks/Benchmarks](http://www.scripps.edu/brooks/Benchmarks)
- [3] For details on the Apo A-1 benchmark, [www.ks.uiuc.edu/Research/apoa1](http://www.ks.uiuc.edu/Research/apoa1)
- [4] J. Harting, S. Wan, P.V. Coveney, *RealityGrid: high-performance computing, visualization, computational steering & teragrids*. *Capability Computing*, 2:4-7, 2003.  
[www.hpcx.ac.uk/about/newsletter/HPCxNews02.pdf](http://www.hpcx.ac.uk/about/newsletter/HPCxNews02.pdf).
- [5] J.C. Phillips, G. Zheng, S. Kumar, L. Kalé, *NAMD biomolecular Simulation on Thousands of Processors*. *Proceedings of the IEEE/ACM SC2002 Conference*. IEEE press, 2002.  
[www.sc-2002.org/paperpdfs/pap.pap277.pdf](http://www.sc-2002.org/paperpdfs/pap.pap277.pdf).

- [6] J. Hein, M. Bull, "*Capability Computing, Achieving Scalability on over 1000 Processors*", Technical report HPCxTR0301  
[www.hpcx.ac.uk/research/hpc/index.html](http://www.hpcx.ac.uk/research/hpc/index.html).
- [7] Luiz DeRose, "*Hardware Performance Monitor (HPM) Toolkit*"  
[www.hpcx.ac.uk/support/documentation/IBMdocuments/HPM.html](http://www.hpcx.ac.uk/support/documentation/IBMdocuments/HPM.html)
- [8] J. Hein, "*Using the Hardware Performance Monitor Toolkit on HPCx*", Technical report HPCxTR0307  
[www.hpcx.ac.uk/research/hpc/technical\\_reports/HPCxTR0307\\_choose.html](http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0307_choose.html)