# Modelling and Predicting Memory Behaviour in Parallel Systems with Network Links—Palladio-based Experiment Report

Philipp Gruber
gruberpp@studi.informatik.uni-stuttgart.de
Uni Stuttgart, Stuttgart, DE

Markus Frank
markus.frank@informatik.uni-stuttgart.de
Uni Stuttgart, Stuttgart, DE

## Abstract

This work improves the capabilities of Palladio to predict the performance of parallel software in multicore environments.

In previous work, we could show that the accuracy of the Palladio simulations is not sufficient for multicore systems. We assume that one reason for this is the memory bandwidth behaviour, which is not included in in the Palladio Component Model and can become a bottleneck in parallel software.

We present an approach to model the memory bandwidth behaviour by the means of an already existing network link concept. We can show that by using network link as a memory model we can improve our predictions up to 26% points using 16 cores on a machine and can receive an accuracy of 90% for our use case.

## 1 Introduction

The Palladio Component Model (PCM) [1] is designed to help software architects to make design decisions in early stages of the development process. Software Architects can make what-if analyses for quality attributes (i.e., response time) based on the software architecture by specifying a user, hardware, and software model.

While the PCM works well for single-core processors, we showed in previous work [1], that Palladio is inaccurate in predicting the performance of multicore systems. In fact, Palladio predicts a linear speedup with an increasing number of cores, which is false.

One factor, we assume has a major impact, is the memory hierarchy and the memory bandwidth behaviour [2]. The memory bandwidth is referring to the capacity limit of the memory bus. If the capacity limit of the bus is reached the cores are subsequently idle, because they have to wait for I/O to continue.

In this paper, we research an approach that uses network links to include the impact and characteristics of the memory bandwidth in the prediction models. Further, we use resource containers in combination with network links to model different cache levels.

To evaluate the approach we use the same use case we used in previous works—a matrix multiplication [1]. In this algorithmic use case, we perform a standardized matrix multiplication, where we multiply matrices of the size 3000x3000 filed with random integers and measure the run-time for different numbers of worker threads and core settings.

We use the previous experiments as a baseline [1] and answer the following two research questions:

- *RSQ1:* Is it possible to model memory behaviour for multicore systems with the network link approach?
- *RSQ2:* Are the predictions more accurate?

This paper is structured as follows, in the modelling chapter we will explain how we modelled the memory behaviour in Palladio and how we calibrated this model. In the execution and measurement chapter, we describe our experiment. In the evaluation, we present our results and answer the research questions. In chapter five we briefly state what lessons we learned. We finish this paper with an outlook where we suggest which topics future work should cover.

## 2 PCM Modeling

The modelling section covers two topics. First, the modelling of the matrix multiplication combined with the memory bandwidth model and second, the calibration of those models, to get reasonable results.

### 2.1 Modeling Creation

**Memtest86 Model:** In a first step, we wanted to evaluate our initial idea to model the memory bandwidth with network links. Therefore, we used a very simple scenario and modelled the behaviour of a memory bandwidth benchmarking tool. For this, we used Memtest86[2], which does loading or writing an increasing size of data from or into the memory. We were able to predict the performance of Memtest86 and used the rough structure of this model in later use cases, (i.e., Figure 1), which we describe in the following in detail.

**Matrix Multiplication Model:** As a starting point for the matrix multiplication model, we used the model provided [1] and extended it by the memory

[1] `https://www.palladio-simulator.com/home/`

[2] `https://www.memtest86.com`

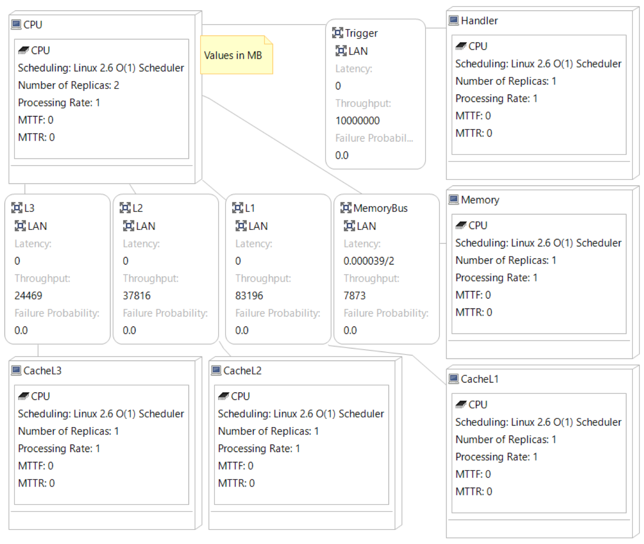

Figure 1: Resource environment view: 2 core system.

model. So here, we only describe the changes made to the model. All used models are made publicly available and can be accessed in our git repository[3].

First of all, we added four memory operation per multiplication operation. This aligns to the behaviour of a matrix multiplication: three reading operations and one writing operation to save the (intermediate) result. That is how we include the traffic of the multiplication to the model.

In the next step, we added network links to our model to reflect the memory bus and additional resource containers to represent RAM, L1, L2 and L3 cache. In Palladio network links can only connect resource containers. Therefore, it was necessary to change the resource environment, allocation and repository diagram. We added for each memory hierarchy level in our hardware (RAM, L1, L2, L3 cache) a resource container. Additionally, we created for each memory hierarchy a dummy component and allocated it to the corresponding resource container. These resource containers were connected to the existing CPU resource container by an individual network element, see Figure 1. In the **S**ervice **Eff**ect Specifications we called the dummy component by an `ExternalCallAction`. This was done before the calculation of the respective memory load operations could be issued. To specify the data over the network we used the property `bytesize`.

In the next section, we will describe how we calibrated the model.

## 2.2 Model Calibration

For a description of how to calibrate the `CPU` resource demand, we refer to our previous work [1]. In the following, we will describe how we calibrated the memory bandwidth, throughput and caching behaviour.

[3]`https://github.com/PhilippGruber/MemBandwidthModels`

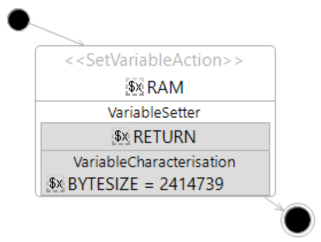

Figure 2: Illustration from Palladio showing the SEFF of the issued network demand.

The throughput of the memory bandwidth was measured with Memtest86 and annotated in the resource environment (see figure 1). Memtest86 is also able to distinguish between load/write operations. The measured throughput values are reflecting the capabilities of the memory bus in our calibration. Since the difference between reading and writing was marginal we took the same mean value for both.

Now we needed to determine the traffic on the memory bus. For this, we can estimate the amount of memory traffic. From above we know that the matrix multiplication loads three values and writes one back. That means we have four memory operation for each loop iteration. These numbers hide how the traffic is distributed over the memory hierarchy. To uncover this fact, we calibrate our model, the cache behaviour, and the memory access numbers with the help of perf [4]. Perf returns the overall numbers of the memory load operations and their distribution over the memory hierarchy. These numbers enable precise modelling in Palladio.

## 3 Execution and Measurements

Unfortunately, we were not able to access the same hardware as used before. Therefore, we had to repeat all experiments, measurements and calibrations again for the following hardware:

### 3.1 Hardware

All experiments are executed on two different machines. The characteristics of the machines are described in Tab. 1.

| Aspect | Machine 1 (M1) | Machine 2 (M2) |
|---|---|---|
| Cores (phys./virt.) | 4x3 / 8x3 | 8x5 / 16x5 |
| Clock Rate | 2.5 GHz | 2.4 GHz |
| L1 cache | 32 KB | 32 KB |
| L2 cache | 256 KB | 256 KB |
| L3 cache | 15 MB | 30 MB |
| RAM | 24 GB | 896 GB |

Table 1: Used hardware

Both machines run on Ubuntu 18.04 with hyper-threading enabled.

[4]`http://man7.org/linux/man-pages/man1/perf.1.html`

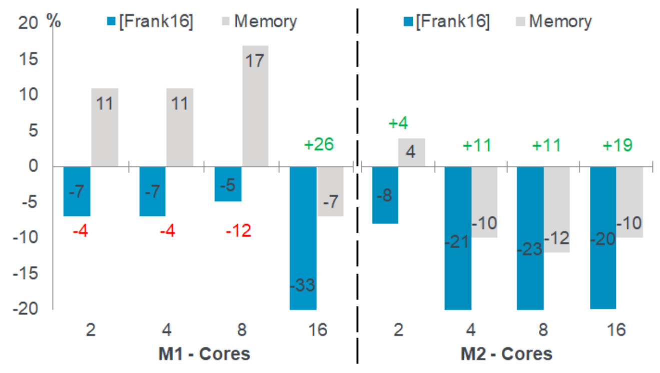

Figure 3: Comparison of simulation accuracy

## 4 Evaluation

In this section we evaluate our results from Sec. 2 and 3 by answering both *RSQ1* and *RSQ2*.

### Evaluation of *RSQ1*

We can affirm that it is possible to model memory behaviour for multicore systems with network links. Once we were able to model Memtest86 in Palladio, we showed this, too. The memtest use case is just loading/writing an increasing size of data from or to the memory (hierarchy), over a limited bus. This is the memory bandwidth and it is possible to adapt this model to our matrix multiplication or any other use-case that aims to include the memory bandwidth/ behaviour.

### Evaluation of *RSQ2*

Figure 3 shows the accuracy of the simulations in percentage. A negative value (i.e., -7%) means that the simulation predicted the run-time to be 7% faster than we measured. We distinguish between the two machines M1 and M2 and compare our approach to the simulations without the memory model in place. For M2 we can report an overall increase of accuracy. For M1 we can only report a great increase for 16 cores.

Reasons for the bad performance on M1 for 2-8 cores, we assume in a badly calibrate caching model for this machine. As soon as we use 16 cores, the system makes use of hyper-threading, which often comes at the cost of cold caches and the simulation accuracy increases at once (still using the memory bandwidth model).

For M2 we can improve the predictions of the simulations by an average of +9% points.

## 5 Lessons Learned

Besides the answer to the research questions, we gained further knowledge, which we will share in the following in the form of lessons learned.

**LL1 Accuracy gain:** First of all, we learned that our approach is actually improving the accuracy in Palladio, as we are able to use the network links to bring the memory bandwidth effect of the memory hierarchy into Palladio.

**LL2 Remaining inaccuracy:** The multicore predictions are still not accurate enough. This suggests the conclusion that we only solved a part of the problem and need to consider additional performance metrics.

**LL3 Better models/estimations:** In the case of M1 using 2-8 cores, our models add too much of a memory bandwidth effect, while for 16 it is not enough. We assume that we are missing other effects as the synchronization of private memory or the contention for shared resources. On the other hand, some parts of our models are overestimating. So for the future, we need to find a better way of calibrating the memory model.

**LL4 Memory behaviour:** We saw that our model is the best when cache behaviour became a bottleneck. In our experiment, this was not the case for M1 2-8 cores. We observed that the cache misses were shrinking with an increasing amount of cores, until the point that the cache size limit was reached. Certainly, it was the case with 16 cores and explained the big accuracy gain. That means all performance models which aim to predict the speed up, should consider such hardware limitations.

**LL5 Core limitations:** Since we noticed that caching works rather well until 8 cores, a repetition of the experiment with more cores (32, 64) will be interesting.

## 6 Conclusion and Outlook

In this short paper, we described an approach on how to model memory bandwidth and hierarchy's in Palladio. We repeated existing experiments to evaluate our approach and could show that we gain additional accuracy of the simulations up to +26% points.

However, we also could show, that memory behaviour is not the only limiting factor to describe the speedup behaviour of parallel software. In future work, we need to investigate different performance-influencing factors to also include them into our prediction models and find better caching models.

## References

[1] M. Frank and M. Hilbrich. "Performance Prediction for Multicore Environments—An Experiment Report". In: *Proceedings of the Symposium on Software Performance 2016, 7-9 November 2016, Kiel, Germany.* 2016.

[2] M. Frank, F. Klinaku, and S. Becker. "Challenges in Multicore Performance Predictions". In: *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering.* ICPE '18. Berlin, Germany: ACM, 2018, pp. 47–48.