

INSTITUT FÜR INFORMATIK
DER LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN



Masterarbeit

**Dynamic PGAS Data
Structures**

Stefan Effenberger



Masterarbeit

Dynamic PGAS Data Structures

Stefan Effenberger

Aufgabensteller: Prof. Dr. Dieter Kranzlmüller

Betreuer: Tobias Fuchs

Abgabetermin: **ADD DATE**

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den **ADD DATE**

.....
(Unterschrift des Kandidaten)

Abstract

ADD ABSTRACT

Inhaltsverzeichnis

1	Introduction	1
1.1	Problem statement	1
1.2	Scope and Objectives	1
2	Background	3
2.1	C++ Concepts	3
2.1.1	Language Concepts	3
2.1.2	Standard Template Library	3
2.2	High Performance Computing	3
2.3	Partitioned Global Address Space	4
2.4	DASH Library	4
3	Related Work	5
3.1	Shared Memory	5
3.1.1	STINGER	5
3.1.2	Ligra	5
3.2	Distributed Memory	5
3.2.1	Parallel Boost Graph Library	5
3.2.2	STAPL Parallel Graph Library	5
4	Graph Container Concepts	7
4.1	Memory Space	7
4.2	Index Space	7
4.3	Iteration Space	7
4.4	Semantics	7
5	Reference Implementation	9
6	Case studies	11
6.1	Static structure	11
6.1.1	Graph traversal	11
6.1.2	Shortest path evaluation	11
6.2	Dynamic Structure	11
6.2.1	Graph partitioning	11
6.2.2	De Bruijn Graph construction	11
7	Evaluation	13
7.1	Micro-benchmarks	13
8	Conclusion	15
8.1	Summary	15

Inhaltsverzeichnis

8.2	Assessment	15
8.3	Outlook	15
	Abbildungsverzeichnis	17
	Literaturverzeichnis	19

1 Introduction

Many scientific projects are largely enabled by simulation. Because such simulations often require huge computational capabilities, single compute nodes with a shared-memory architecture cannot provide enough computation power and storage for numerous cases. For this reason, in High Performance Computing (HPC), work is distributed among multiple, interconnected nodes to facilitate the solving of large problems in a timely manner. Since processors cannot directly access the memory of other nodes, the traditional programming model for such systems requires programmers to explicitly distribute data between nodes via message passing. This imposes high demands on the programming skills of scientists who might not have a background in computer science.

Therefore, with the Partitioned Global Address Space (PGAS) programming model, a new approach is proposed: The memory space of individual nodes in a system is unified within a global address space so that each node can directly access the memory of all other nodes. Programmers are still required to keep data access between nodes to a minimum because data transferal over an interconnect is costly. To further reduce the demands on the programmer, distributed data structures that handle data distribution and load balancing are needed.

Furthermore, data-intensive tasks have been gaining a continually growing interest in the scientific community. Traditionally, applications in HPC follow a computation-centric approach by solving numerical algorithms in the fastest possible way. As “Big Data” is becoming increasingly important in scientific projects, a shift towards more data-oriented applications can be observed in recent HPC projects [ZZZ⁺14]. This trend requires distributed data structures that allow for the storage of large amounts of irregular data and cater to the needs of ever-changing dynamic data.

1.1 Problem statement

1.2 Scope and Objectives

2 Background

This chapter covers some fundamental background knowledge needed for a better understanding of the following chapters of this thesis. Only explanations directly relevant to the topics of this thesis are provided.

Since the result of this work is a C++ concept, some important language expressions and concepts are firstly discussed, along with a description of the Standard Template Library on which concepts this work is built upon. The reader is then introduced to the domain of High Performance Computing which is the main application area for this work. A brief overview of the Partitioned Global Address Space programming model is then followed by a description of the DASH Library which provides core concepts used in this thesis.

2.1 C++ Concepts

2.1.1 Language Concepts

2.1.2 Standard Template Library

2.2 High Performance Computing

High Performance Computing (HPC) is a broad term describing advances for the fastest possible computation of a given problem. Gustafson's Law [Gus88] suggests that a compute system can linearly grow with the problem size: A problem of two times its original size can be computed on a system with twice as many processors in the same time (best case scenario). This means that very large problems can be computed in an acceptable timeframe if there is a sufficiently large compute system available. Depending on the problem size, two different system architectures are used in HPC:

Shared Memory A shared memory system consists of a single node with multiple processors connected to the same random access memory. Memory access for the different processors can be uniform, but many systems implement a non-uniform memory access (NUMA) design where a part of the memory is assigned to each of the processors. A processor in a NUMA system can access its assigned memory faster than the memory of the other processors. Because processors can access all data at all times, communication between processors has a low cost which simplifies programming on these systems in comparison to distributed memory systems. Achieving high performance on NUMA systems is more problematic because the programmer has to take data locality into account [Lam13].

Distributed Memory Multi-processor systems in which each processor has access to its own memory space are called distributed memory systems. These systems usually consist of several shared memory nodes with the processors of one node not being able to directly

access memory of other nodes. While single shared memory systems can only be scaled to a certain extent, the scalability of distributed systems is much higher [PTM96].

The nodes are connected with a network interconnect for communication between the processors. Due to the latency of the interconnect being significantly higher than the latency of a memory bus in a shared memory system, communication is much more costly. This imposes higher demands on the programmers' skills in comparison to shared memory systems.

The largest problems in science are computed on “supercomputers” like the *SuperMUC* at the *Leibniz Rechenzentrum* in Munich. These distributed memory machines consist of hundreds or even thousands of homogeneous nodes that are connected with a specialized interconnect. To this date, *message passing* is the prevalent programming model for such systems.

2.3 Partitioned Global Address Space

2.4 DASH Library

3 Related Work

3.1 Shared Memory

3.1.1 STINGER

3.1.2 Ligra

3.2 Distributed Memory

3.2.1 Parallel Boost Graph Library

3.2.2 STAPL Parallel Graph Library

4 Graph Container Concepts

4.1 Memory Space

4.2 Index Space

4.3 Iteration Space

4.4 Semantics

5 Reference Implementation

6 Case studies

6.1 Static structure

6.1.1 Graph traversal

6.1.2 Shortest path evaluation

6.2 Dynamic Structure

6.2.1 Graph partitioning

6.2.2 De Bruijn Graph construction

7 Evaluation

7.1 Micro-benchmarks

8 Conclusion

8.1 Summary

8.2 Assessment

8.3 Outlook

Abbildungsverzeichnis

Literaturverzeichnis

- [Gus88] GUSTAFSON, John L.: Reevaluating Amdahl's Law. In: *Commun. ACM* 31 (1988), Mai, Nr. 5, 532–533. <http://dx.doi.org/10.1145/42411.42415>. – DOI 10.1145/42411.42415. – ISSN 0001–0782
- [Lam13] LAMETER, Christoph: NUMA (Non-Uniform Memory Access): An Overview. In: *Queue* 11 (2013), Juli, Nr. 7, 40:40–40:51. <http://dx.doi.org/10.1145/2508834.2513149>. – DOI 10.1145/2508834.2513149. – ISSN 1542–7730
- [PTM96] PROTIC, J. ; TOMASEVIC, M. ; MILUTINOVIC, V.: Distributed shared memory: concepts and systems. In: *IEEE Parallel Distributed Technology: Systems Applications* 4 (1996), Summer, Nr. 2, S. 63–71. <http://dx.doi.org/10.1109/88.494605>. – DOI 10.1109/88.494605. – ISSN 1063–6552
- [ZZZ⁺14] ZHAO, D. ; ZHANG, Z. ; ZHOU, X. ; LI, T. ; WANG, K. ; KIMPE, D. ; CARNS, P. ; ROSS, R. ; RAICU, I.: FusionFS: Toward supporting data-intensive scientific applications on extreme-scale high-performance computing systems. In: *2014 IEEE International Conference on Big Data (Big Data)*, 2014, S. 61–70