



Characterizing Load and Communication Imbalance in Parallel Applications

David Böhme

Forschungszentrum Jülich GmbH
Institute for Advanced Simulation (IAS)
Jülich Supercomputing Centre (JSC)

Characterizing Load and Communication Imbalance in Parallel Applications

David Böhme

Schriften des Forschungszentrums Jülich

IAS Series

Volume 23

ISSN 1868-8489

ISBN 978-3-89336-940-9

Contents

1	Introduction	1
1.1	Parallel Computers	1
1.1.1	Hardware Architectures	2
1.1.2	Programming Models	3
1.1.3	Execution Models	6
1.1.4	Parallel Performance	6
1.2	Performance Analysis	7
1.2.1	Objectives and Requirements	7
1.2.2	Performance-Analysis Techniques	8
1.2.3	Performance Tools	11
1.3	Contribution of this Thesis	12
2	Event-Trace Analysis	17
2.1	Event-Tracing Applications	17
2.2	Scalasca	18
2.3	The Scalasca Event Model	20
2.3.1	MPI Events	20
2.3.2	OpenMP Events	23
2.4	Wait-state Classification in Scalasca	23
2.4.1	Wait States in MPI Communication	23
2.4.2	Wait States in OpenMP Synchronization	26
2.5	Trace Analysis Methodology	26
2.6	Tracing Constraints and Requirements	28
3	Characterizing Imbalance	31
3.1	Execution Model	31
3.2	Load and Communication Imbalance	32
3.3	Patterns of Imbalance	34
3.3.1	Experiment Setup	34
3.3.2	Distribution across Processes	34
3.3.3	Interference between Imbalances	38
3.3.4	Transformation over Time	38
3.3.5	Imbalance in MPMD Programs	39
3.4	Causes of Imbalance	40
3.4.1	Program-Internal Causes	40
3.4.2	Program-External Causes	41
3.5	Eliminating Imbalance	42

3.6	Related Work	43
3.7	Requirements for Imbalance Analysis	45
4	Identifying Root Causes of Wait States	49
4.1	A Wait-State Formation Model	49
4.1.1	Wait States	50
4.1.2	Delay	51
4.1.3	Costs	51
4.2	Delay Cost Calculation	52
4.2.1	Program Model	52
4.2.2	Synchronization Points and Wait States	52
4.2.3	Synchronization Intervals	53
4.2.4	Causes of Wait States	54
4.2.5	Delay Costs	55
4.2.6	Wait-state Propagation Characteristics	57
4.3	A Scalable Delay Analysis Approach	58
4.3.1	Trace Replay	58
4.3.2	Delay Detection	59
4.3.3	Visual Representation of Delay Costs	63
4.4	Evaluation	65
4.4.1	Zeus-MP/2	65
4.4.2	CESM Sea Ice Model	68
4.4.3	Illumination	69
4.5	Related Work	72
5	Critical-Path Based Performance Analysis	77
5.1	Revisiting Critical-Path Analysis	77
5.2	Critical-Path Analysis Concept	78
5.2.1	The Critical Path	79
5.2.2	Critical-Path Profile	79
5.2.3	Critical-Path Imbalance Indicator	80
5.2.4	Performance-Impact Indicators	81
5.3	Scalable Critical-Path Detection	84
5.4	Evaluation	86
5.4.1	PEPC	86
5.4.2	ddcMD	87
5.5	Related Work	89
6	Comparative Study	91
6.1	Functional Aspects	91
6.1.1	Functional Comparison	91
6.1.2	Suitability	92
6.2	Technical Aspects	95
6.2.1	Trace-Analysis Applicability	95
6.2.2	Scalability	98

6.2.3 Limitations	99
7 Conclusion & Outlook	103

The amount of parallelism in modern supercomputers currently grows from generation to generation. Further application performance improvements therefore depend on software-managed parallelism: the software must organize data exchange between processing elements efficiently and optimally distribute the workload between them. Performance analysis tools help developers of parallel applications to evaluate and optimize the parallel efficiency of their programs. This dissertation presents two novel methods to automatically detect imbalance-related performance problems in MPI programs and intuitively guide the performance analyst to inefficiencies whose optimization promise the highest benefit.

The first method, the delay analysis, identifies the root causes of wait states. A delay occurs when a program activity needs more time on one process than on another, which leads to the formation of wait states at a subsequent synchronization point. Wait states are the primary symptom of load imbalance in parallel programs. While wait states themselves are easy to detect, the potentially large temporal and spatial distance between wait states and the delays causing them complicates the identification of wait-state root causes.

The delay analysis closes this gap, accounting for both short-term and long-term effects. The second method is based on the detection of the critical path, which determines the effect of imbalance on program runtime. The critical path is the longest execution path in a parallel program without wait states: optimizing an activity on the critical path will reduce the program's runtime. Comparing the duration of activities on the critical path with their duration on each process yields a set of novel, compact performance indicators. These indicators allow users to evaluate load balance, identify performance bottlenecks, and determine the performance impact of load imbalance at first glance by providing an intuitive understanding of complex performance phenomena.

Both analysis methods leverage the scalable event-trace analysis technique employed by the Scalasca toolset: by replaying event traces in parallel, the bottleneck search algorithms can harness the distributed memory and computational resources of the target system for the analysis, allowing them to process even large-scale program runs.

This publication was written at the Jülich Supercomputing Centre (JSC) which is an integral part of the Institute for Advanced Simulation (IAS). The IAS combines the Jülich simulation sciences and the supercomputer facility in one organizational unit. It includes those parts of the scientific institutes at Forschungszentrum Jülich which use simulation on supercomputers as their main research methodology.