Optimizing the execution of many-task computing applications using in-memory distributed file systems
Uta, A.

2017

document version
Publisher's PDF, also known as Version of record

Link to publication in VU Research Portal

citation for published version (APA)

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:
vuresearchportal.ub@vu.nl

Download date: 10. Mar. 2024
1 General Introduction
1.1 Many-Task Computing .......................... 2
1.2 MTC Application Storage Requirements .................. 3
1.3 In-Memory Computing .......................... 3
1.4 Related Approaches .......................... 5
1.5 Thesis Contributions and Outline .................. 6
1.5.1 Chapter 2: MemFS - The No-Locality Approach ........ 7
1.5.2 Chapter 3: MemEFS - Elasticity Improves Resource Utilization 7
1.5.3 Chapter 4: MemEFS - Network-Awareness ........ 8
1.5.4 Chapter 5: MemFSS - Memory Scavenging ........ 8

2 Overcoming Data Locality: an In-Memory Runtime File System with Symmetrical Data Distribution 11
2.1 Introduction .................................. 12
2.2 Background and Related Work .................. 13
2.3 System Design .................................. 16
2.3.1 Components .................................. 16
### 2.3.2 Implementation Building Blocks

2.4 Evaluation

- 2.4.1 MTC Envelope Evaluation
- 2.4.2 Application Benchmarks & Multicore Scalability
- 2.4.3 Summary & Discussion

2.5 Conclusions and Future Work

---

#### 3 MemEFS: an Elastic In-Memory Runtime File System for eScience Applications

3.1 Introduction

3.2 Background

- 3.2.1 Elasticity Requirements

3.3 MemEFS Design and Implementation

- 3.3.1 Two-Layer Hashing Scheme
- 3.3.2 Load Balancing
- 3.3.3 Initialization and Reconfiguration
- 3.3.4 Implementation

3.4 Evaluation

- 3.4.1 Experimental Setup
- 3.4.2 Preliminary Results
- 3.4.3 Elastic Scaling Policies
- 3.4.4 MemEFS Performability
- 3.4.5 Storage Load Balancing
- 3.4.6 Discussion

3.5 Related Work

- 3.5.1 Elasticity in Data Storage Systems
- 3.5.2 Load Balancing Schemes in DHTs

3.6 Conclusion and Future Work

---

#### 4 MemEFS - a Network-Aware In-Memory Distributed File System

4.1 Introduction

4.2 MemEFS Network Adaptation

- 4.2.1 MemEFS
- 4.2.2 Network Awareness

4.3 Network Adaptability Evaluation

- 4.3.1 Cloud Experimental Setup
- 4.3.2 Network Adaptability Experiments
- 4.3.3 Discussion

4.4 Related Work

4.5 Conclusions
5 Towards Resource Disaggregation - Memory Scavenging for Scientific Workloads

5.1 Introduction .................................................. 82
5.2 Motivation and Background ................................. 83
  5.2.1 The Achieved Parallelism of Scientific Workflows .... 83
  5.2.2 Cluster Resource Utilization ............................ 84
  5.2.3 MemFS .................................................. 85
5.3 MemFSS Design ................................................. 85
  5.3.1 The MemFSS System ..................................... 85
  5.3.2 Hashing .................................................. 86
  5.3.3 POSIX (FUSE) Interface ................................. 87
  5.3.4 Data & Metadata Placement ............................. 87
  5.3.5 Fault-Tolerance ......................................... 88
  5.3.6 Data Privacy & Isolation ............................... 88
5.4 Evaluation ..................................................... 89
  5.4.1 Experimental Setup ..................................... 89
  5.4.2 Scavenging Overhead Baseline ......................... 90
  5.4.3 Real-World Application Slowdown ...................... 93
  5.4.4 Resource Consumption Reduction ...................... 97
  5.4.5 Summary ................................................ 97
5.5 Related Work .................................................. 98
  5.5.1 Cycle Scavenging - Volunteer Computing ............... 98
  5.5.2 Remote Memory Usage .................................. 99
  5.5.3 Weighted Hashing ....................................... 99
5.6 Conclusions .................................................. 100

6 General Conclusions ............................................. 101
  6.1 Thesis Contributions ...................................... 102
  6.2 Future Directions .......................................... 103

References ......................................................... 105

Summary .......................................................... 117

Samenvatting ...................................................... 121