An Elastic Layers Pattern Approach with Dynamically Added Layers

Christian Zirkelbach and Marc Adolf

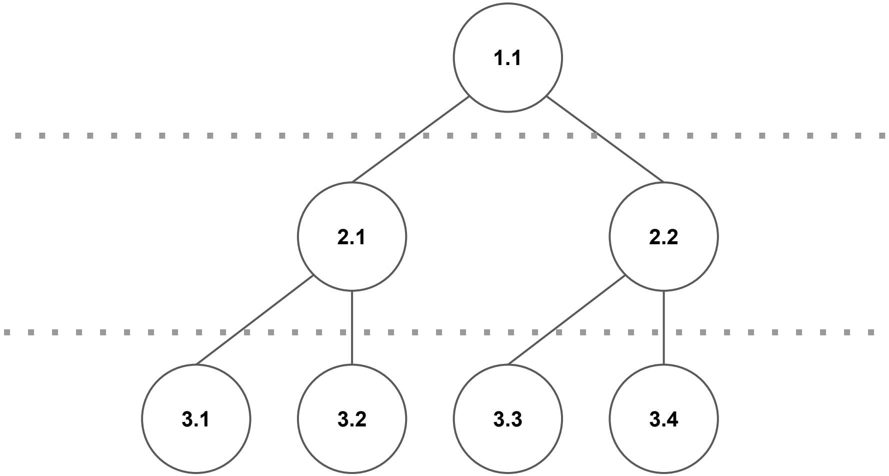
09.11.2016

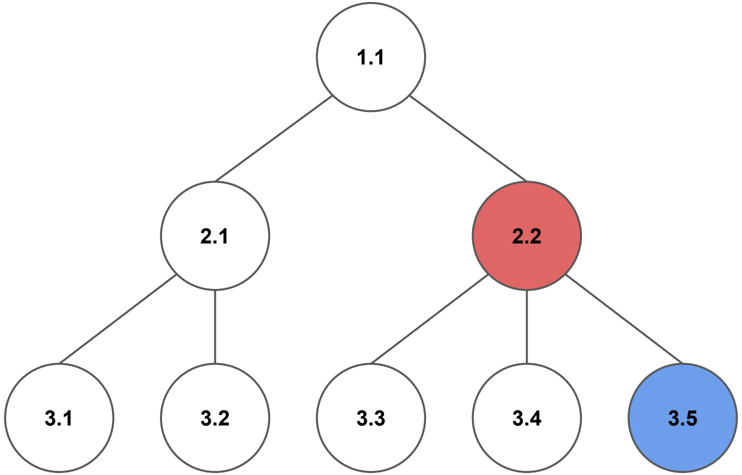
C A U
Christian-Albrechts-Universität zu Kiel
Technische Fakultät

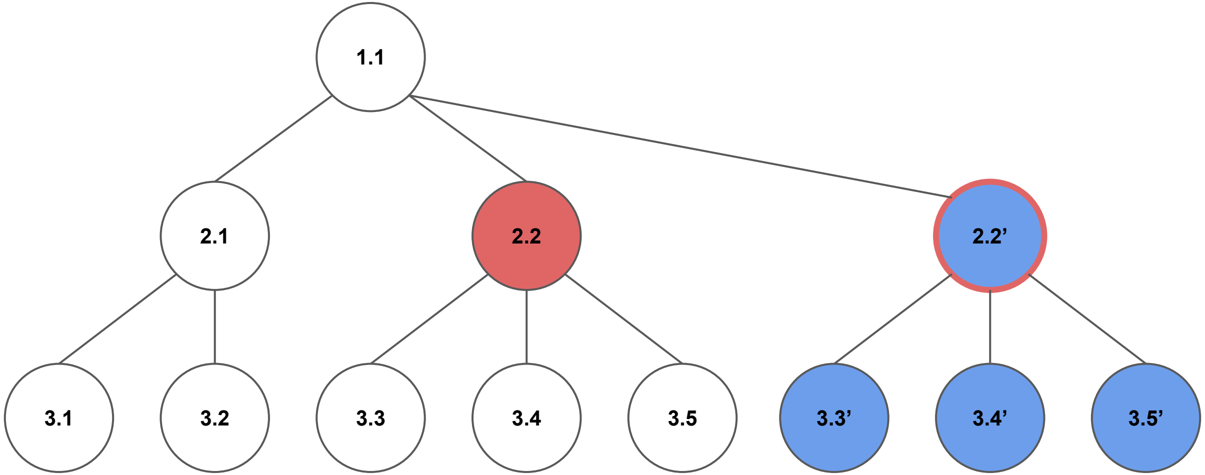
Motivation

- Monitoring of applications in ExplorViz [3]
- Different filter and analyzing steps for probed data
- Resulting architecture is a (parallel) layered architecture [2]
- Existing scaling operations (in this architecture) not sufficient

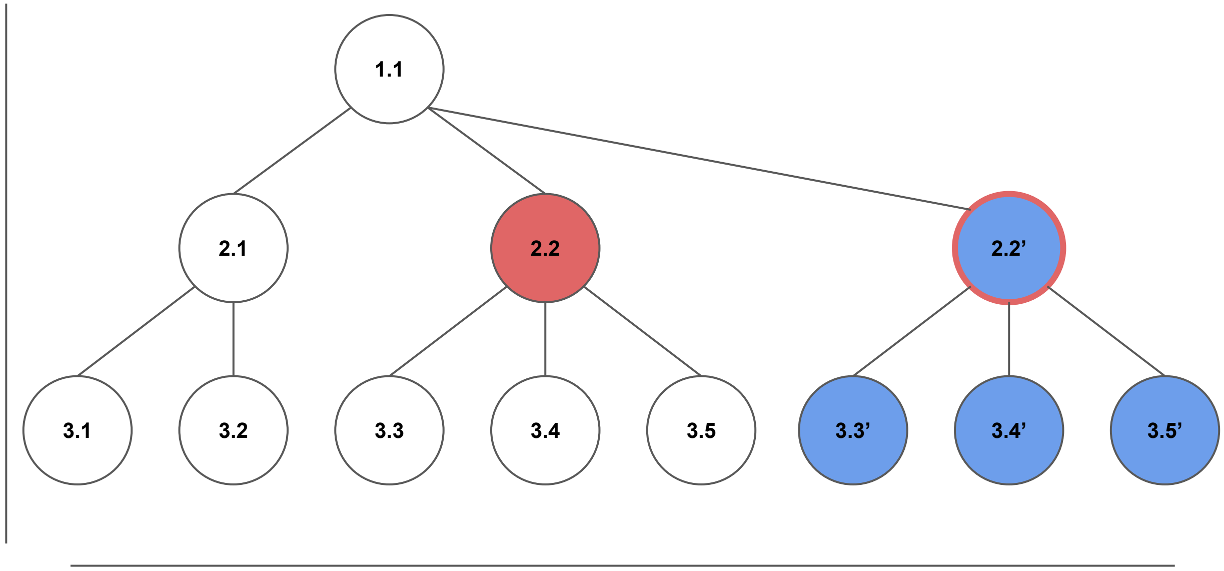
Parallel Layers Pattern [2] (Foundation)


1.1
2.1
2.2
3.1
3.2
3.3
3.4


1.1
2.1
2.2
3.1
3.2
3.3
3.4
3.5

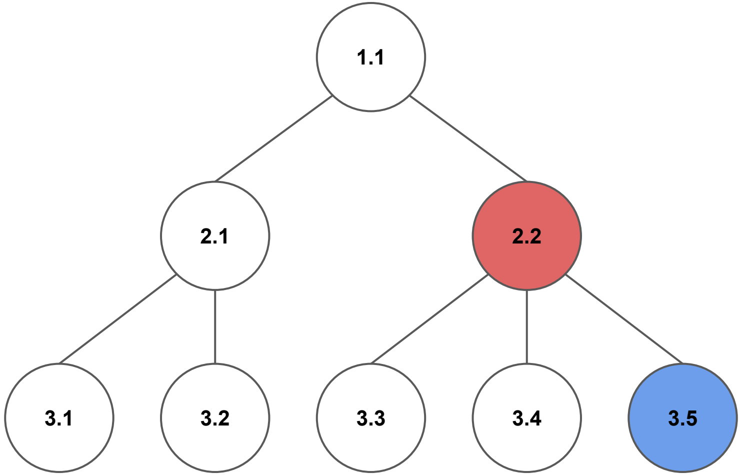

1.1
2.1
2.2
2.2’
3.1
3.2
3.3
3.4
3.5
3.3’
3.4’
3.5’

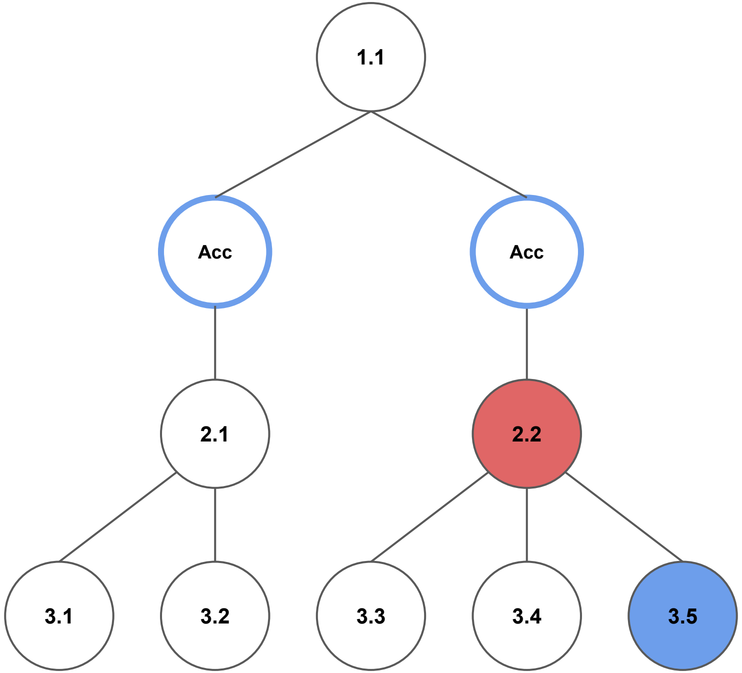
Height:
Fixed by
system design/
architecture

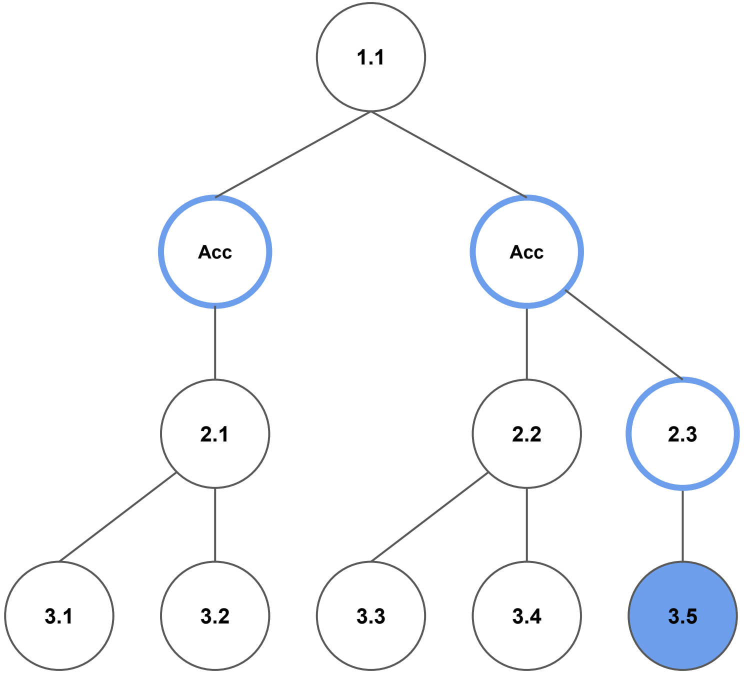


Width: Scalable at runtime

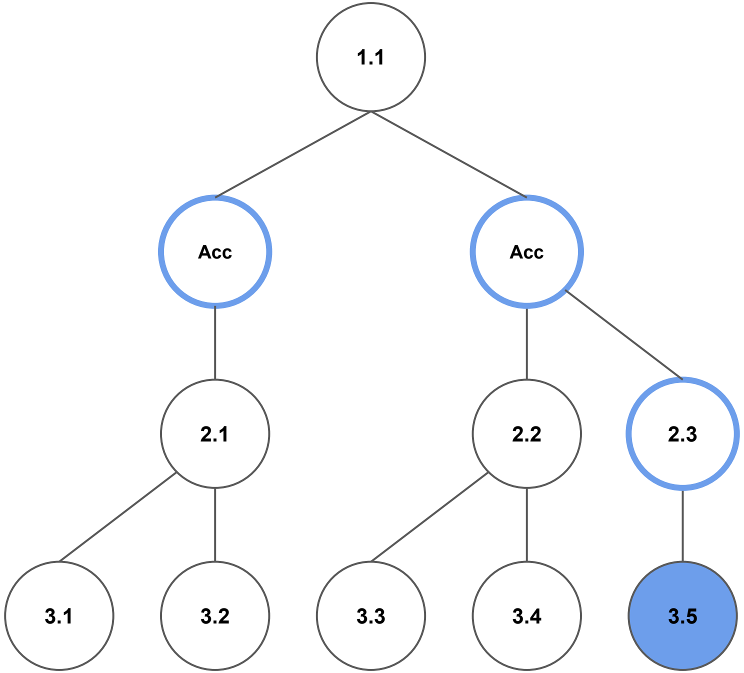
Our Dynamic Height Approach for the Parallel Layers Pattern


1.1
2.1
2.2
3.1
3.2
3.3
3.4
3.5


1.1
Acc
Acc
2.1
2.2
3.1
3.2
3.3
3.4
3.5

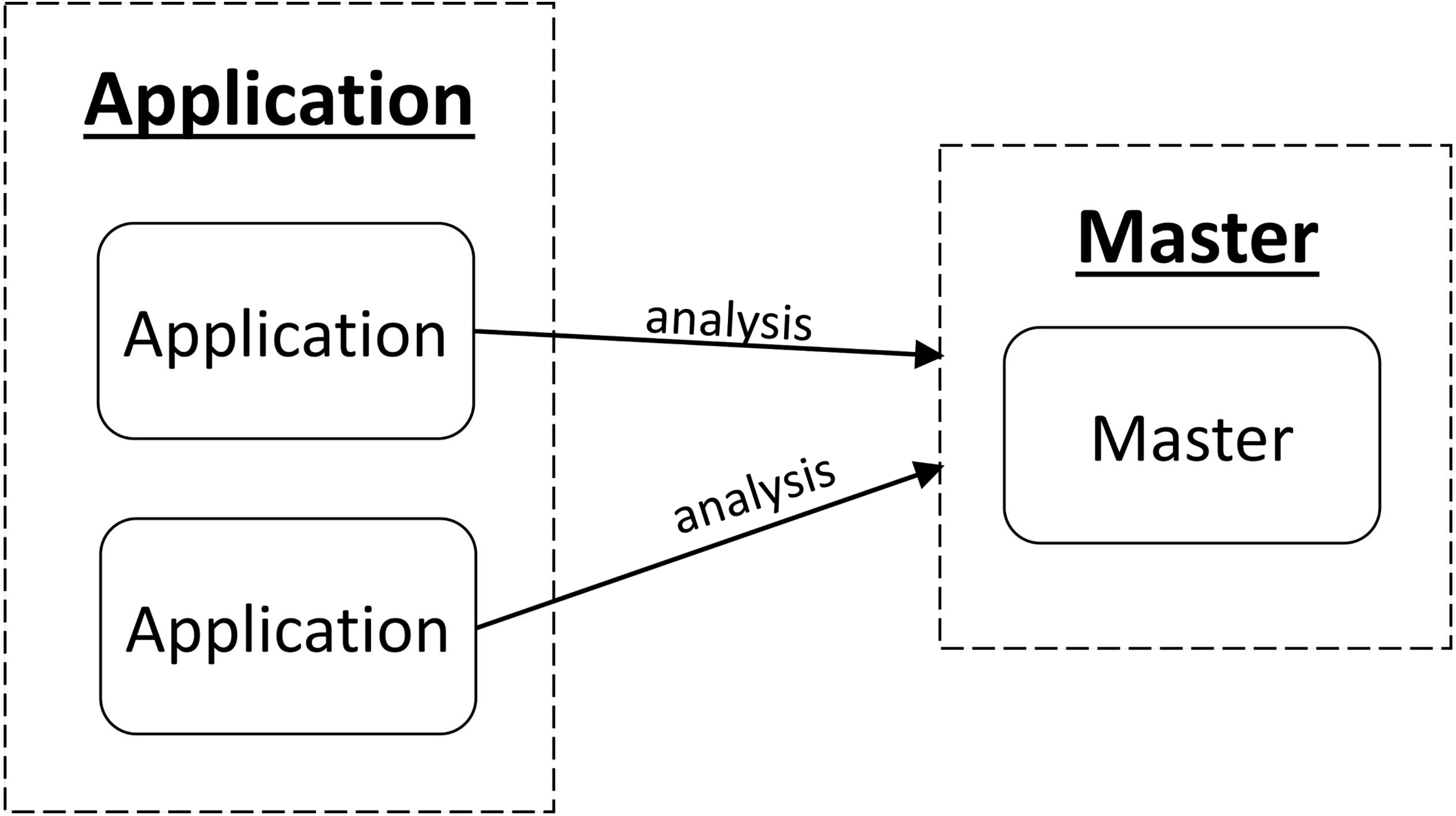

1.1
Acc
Acc
2.1
2.2
2.3
3.1
3.2
3.3
3.4
3.5

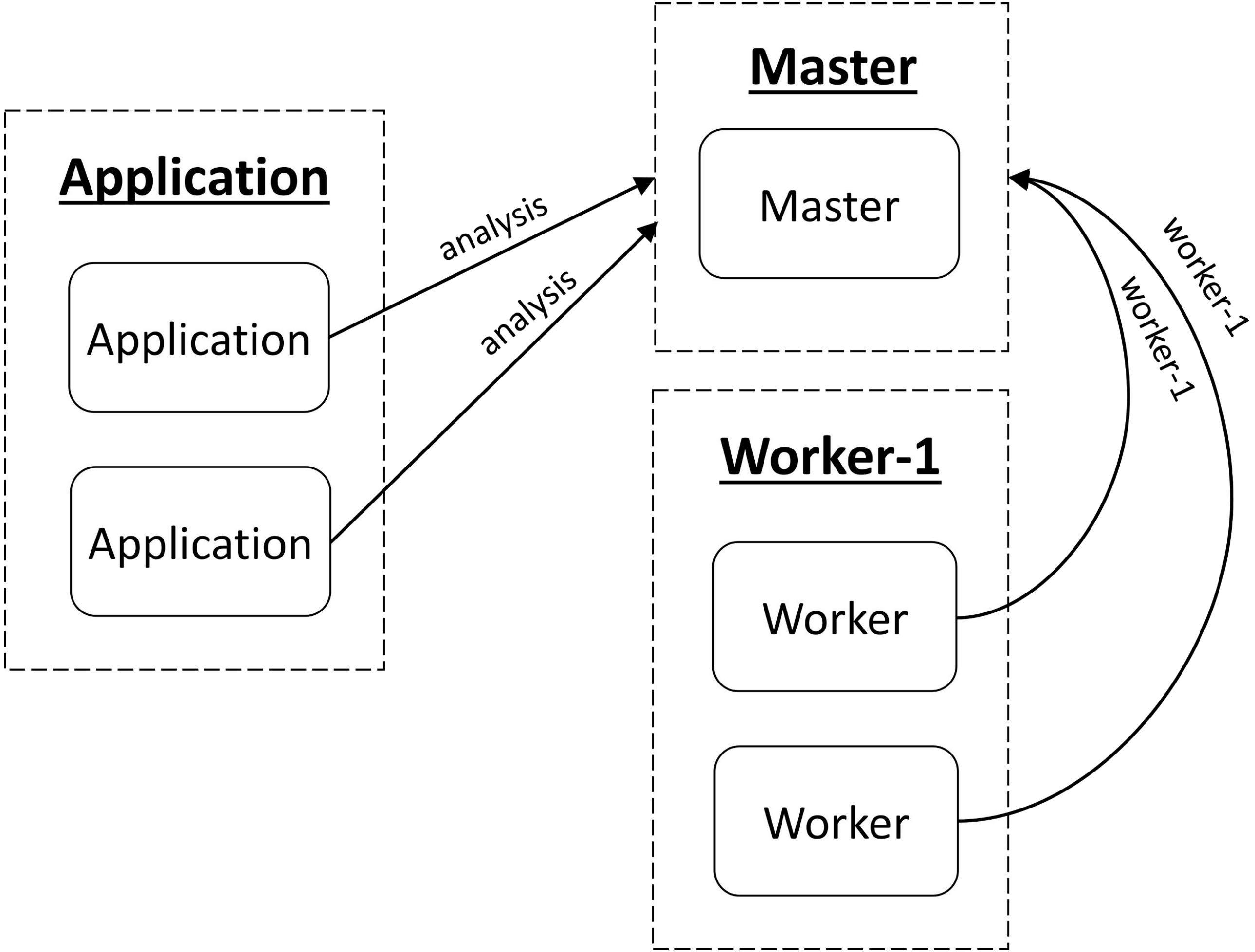
Height:
Scalable at runtime.
Accumulator
defined at design
time

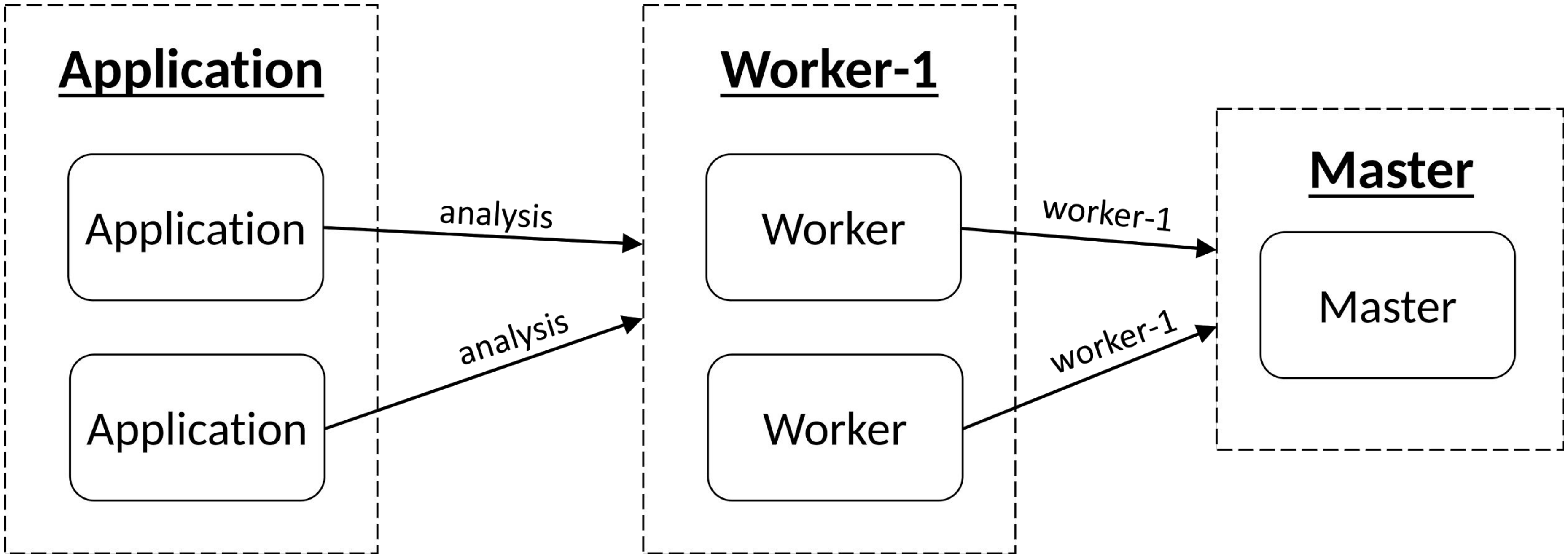


Width: (Still) Scalable at runtime

Implementation in ExplorViz


Application
Application
Application
analysis
analysis
Master
Master


Application
Application
Application
analysis
analysis
Master
Master
Worker-1
Worker
Worker
worker-1
worker-1


Application
Application
Application
analysis
analysis
Worker-1
Worker
Worker
worker-1
worker-1
Master
Master

Evaluation

Evaluation Setup

- Private cloud running OpenStack containing seven servers
- 224 VCPU cores (112 real cores) and 896 GB of RAM
- Object system: web application JPetStore
- Flavor used for every dynamically started instance

  (Master,Worker, and JPetStore nodes): 1 VCPU, 3 GB of RAM
- First presented at ESOCC, 2015 [1]

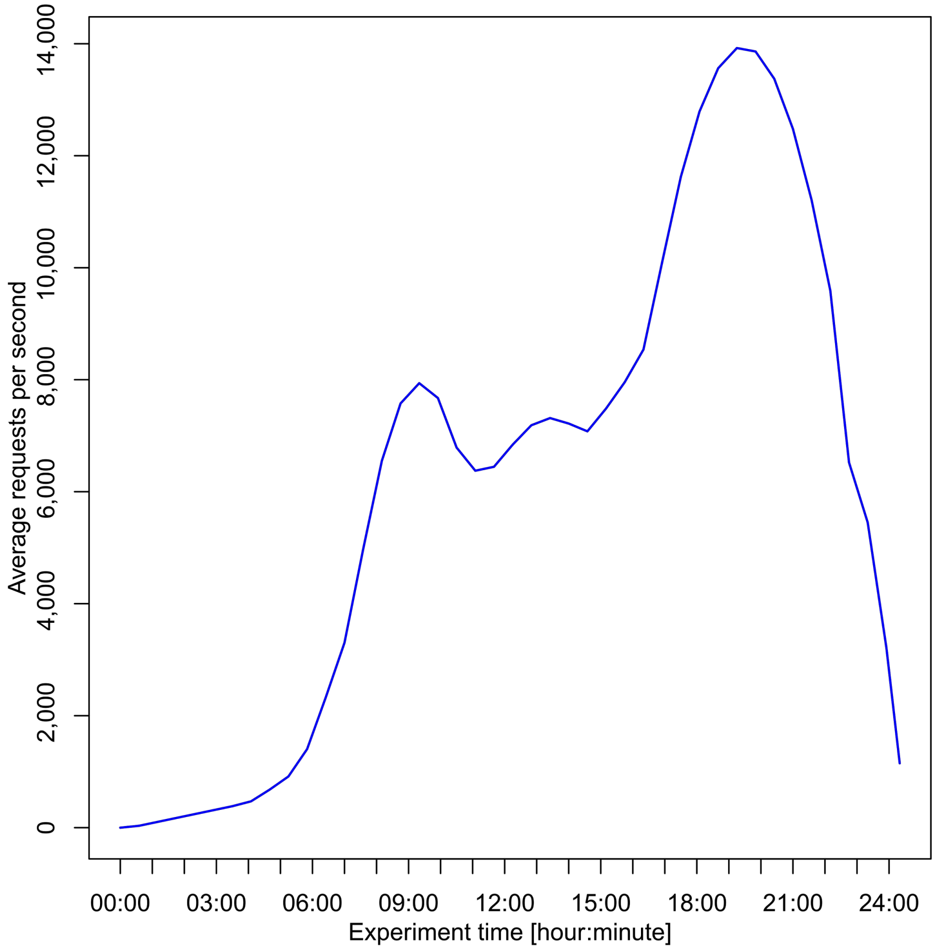


Fig. 1: Employed workload curve [1]

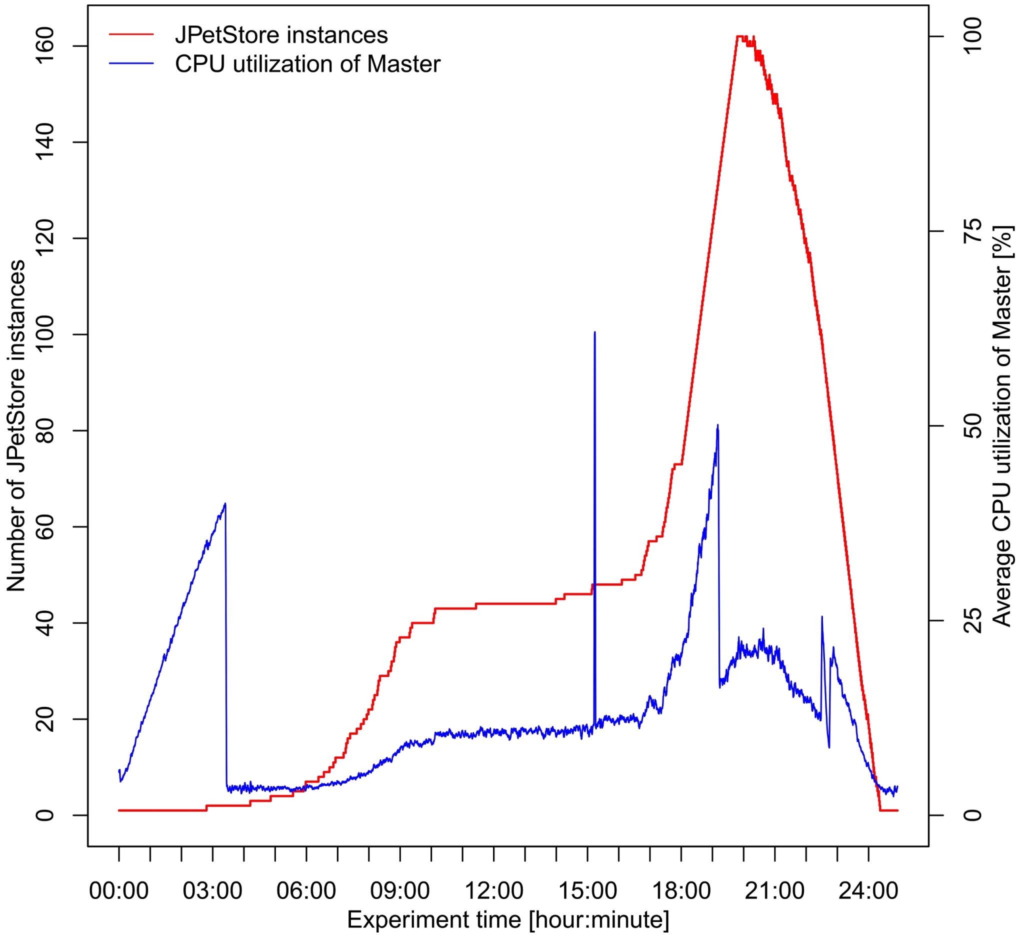


Fig. 2: JPetStore instance count and average CPU utilization of Master node [1]

Conclusion and Outlook

- Extension to an existing Parallel Layers Pattern [2]
- Increased level of scalability
- Successful integration and evaluation in ExplorViz[1]
  - Fluctuating workloads are handled dynamically
  - Open source and replication package provided

ExplorViz

[1] www.explorviz.net

References

[1] F. Fittkau and W. Hasselbring. “Elastic Application-Level Monitoring for Large Software Landscapes in the Cloud.” In: Proceedings of ESOCC. Springer, 2015.

[2] J. L. Ortega-Arjona. “The Parallel Layers Pattern. A Functional Parallelism Architectural Pattern for Parallel Programming.” In: Proceedings of SugarLoafPLoP. 2007.

[3] F. Fittkau, A. Krause, and W. Hasselbring. “Software landscape and application visualization for system comprehension with ExplorViz.” In: Information and Software Technology (2016). http://dx.doi.org/10.1016/j.infsof.2016.07.004.

[4] N. Huber et al. “Model-Based Autonomic and Performance-Aware System Adaptation in Heterogeneous Resource Environments: A Case Study.” In: Cloud and Autonomic Computing (ICCAC). 2015.

[5] S. Newman. Building Microservices. O’Reilly Media, Inc., 2015.

[6] J. Dean and S. Ghemawat. “Mapreduce: simplified data processing on large clusters.” In: Proceedings of OSDI. 2004.

Threats to Validity

- Only one environment and one application
- Only two worker levels (due to only 216 possible instances)
- Similar traces generated by JPetStore

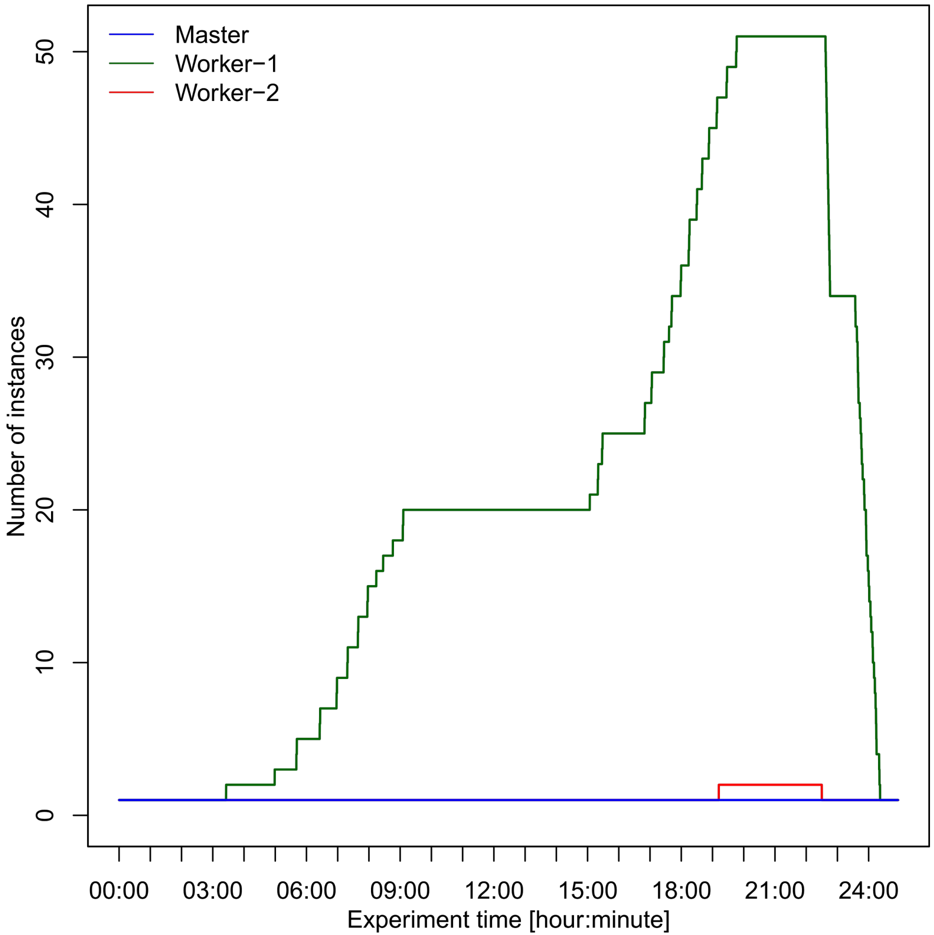


Fig. 3: Analysis nodes and number of instances in each level [1]

Related Work

- Architectural Approaches:
    - Microservices [5]
    - Mapreduce [6]

- Self-Adapting Techniques like [4].