pTA: A Programmable Teaching Assistant for Lab Courses

Jawad Tahir
Technical University of Munich
jawad.tahir@tum.de

Raj Mandal
Technical University of Munich
raj.mandal@tum.de

Olha Stefanova
Technical University of Munich
olha.stefanova@tum.de

Hans-Arno Jacobsen
University of Toronto
jacobsen@eecg.toronto.edu

Christoph Doblander
Technical University of Munich
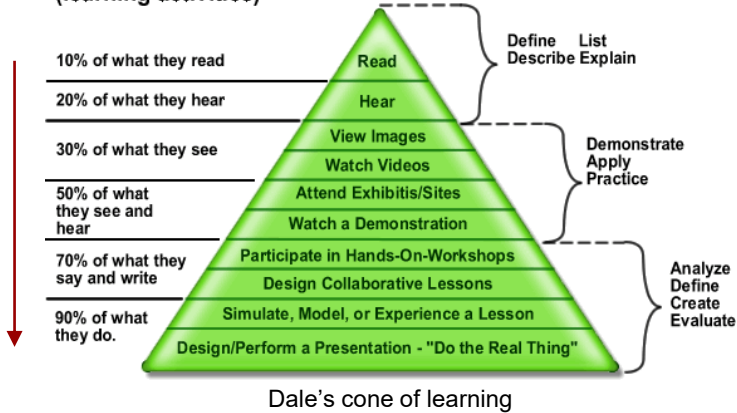christoph.doblander@tum.de

Ruben Mayer
University of Bayreuth
ruben.mayer@uni-bayreuth.de

Lab course - Intro

People generally remember...
(learning activities)

People are able to...
(learning outcomes)

- 10% of what they read — Read — Define List Describe Explain
- 20% of what they hear — Hear
- 30% of what they see — View Images — Demonstrate Apply Practice
- Watch Videos
- 50% of what they see and hear — Attend Exhibitis/Sites
- Watch a Demonstration
- 70% of what they say and write — Participate in Hands-On-Workshops — Analyze Define Create Evaluate
- Design Collaborative Lessons
- 90% of what they do. — Simulate, Model, or Experience a Lesson
- Design/Perform a Presentation - "Do the Real Thing"

Dale's cone of learning

VectorStock®   VectorStock.com/20300366

- Lab courses are an essential part of data systems education
- Need to apply the concepts taught in order to really understand them deeply

- But lab courses induce high workload for instructors / Tas
- Scaling them is difficult

Lab course - Cloud Databases (CDB)

- Offered at TUM
- Students incrementally design and develop a replicated distributed data store (key-value store)
- Divided in 4+1 time-barred milestones (MS)
  - The final MS is students' extension
- Students work on MSs and submit their systems for evaluation
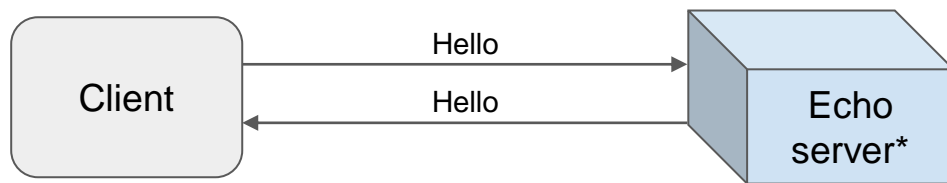 - A Gitlab instance is used to collaborate

https://cdb.dis.cit.tum.de

Evaluation
- Course staff (instructor, TAs) run various tests on submissions to verify the implementation
- Grades are awarded as per the results of tests

VectorStock®   VectorStock.com/36367394

Course synopsis - MS1

Client

Hello

Hello

Echo server*

Learning outcomes:
- Network communication
- Standard streams
- Containerization
- De/serialization

*Provided

Course synopsis - MS2

Client

put K V

success
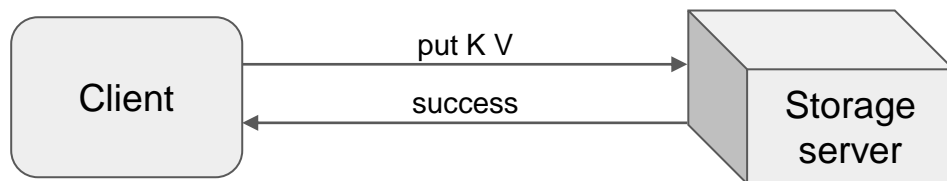
Storage
server
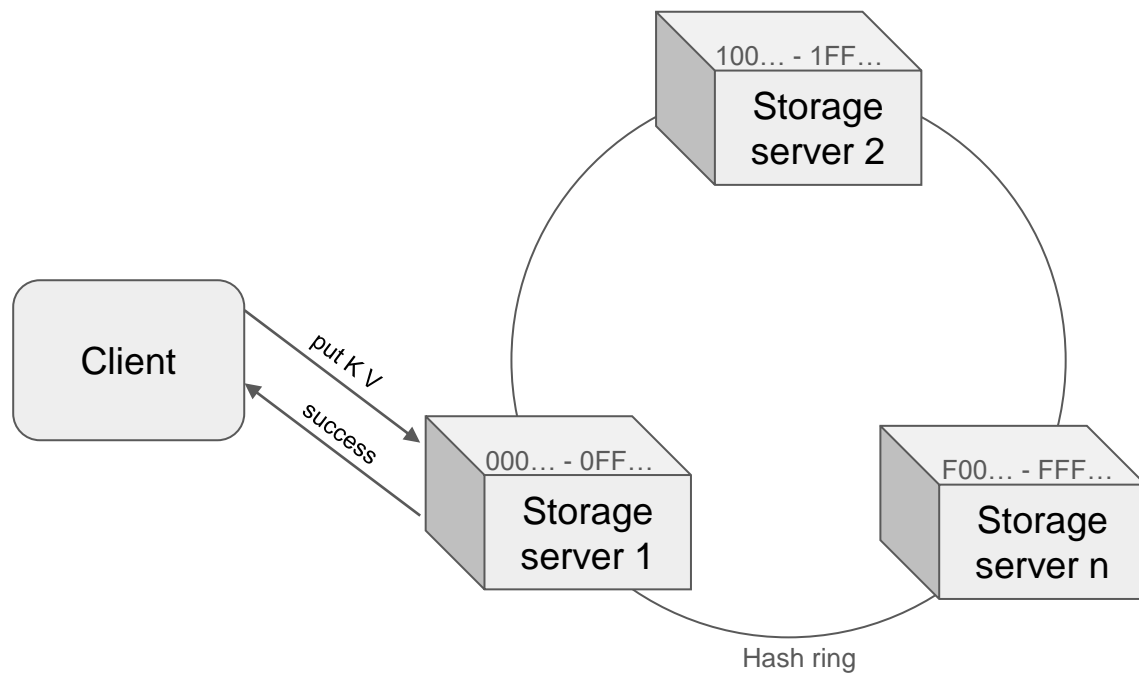
Learning outcomes:
- Persistent storage
- Server development
- Multithreading
- Caching
- Efficient data storage
  and retrieval algorithms

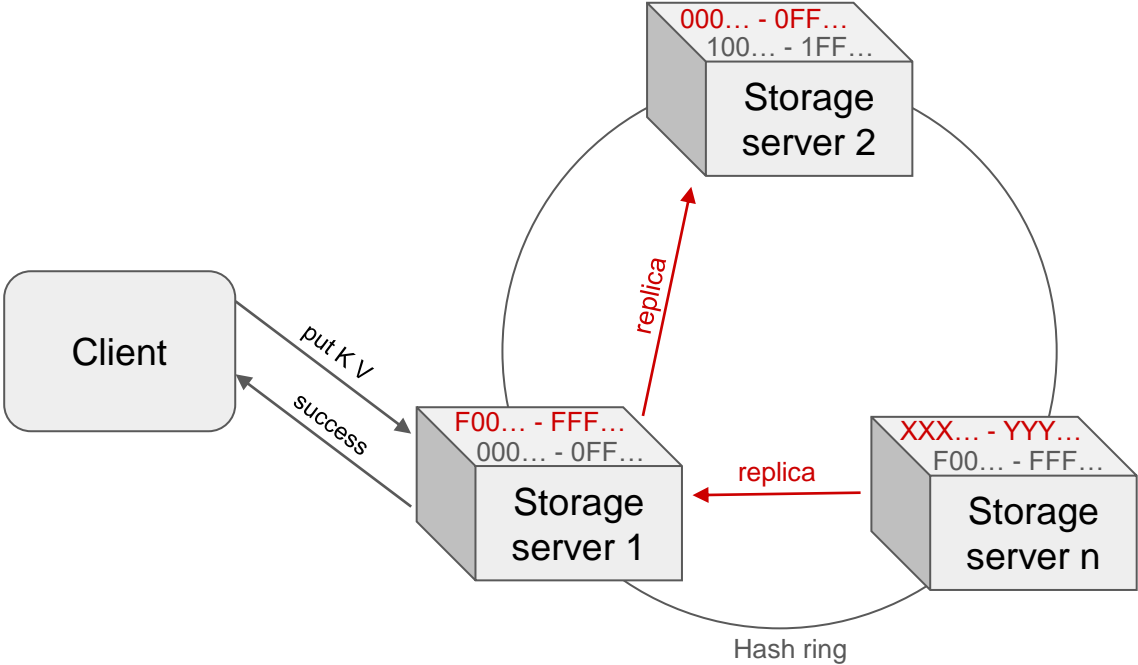# Course synopsis - MS3

Learning outcomes:
- Distributed storage
- Hashing
- Hash ring
- Peer-to-peer systems
- Load balancing

Course synopsis - MS4

Learning outcomes:
- Replicated distributed storage system
- Replication strategies
- Content distribution
- Fault detection and recovery
- Benchmarking

Evaluation - Pain points

Fair evaluation

Frequent visits to the course staff => increased workload

Slow and time-consuming process

Our approach: Programmable Teaching Assistant (pTA)

- A framework to automate testing of students' solutions
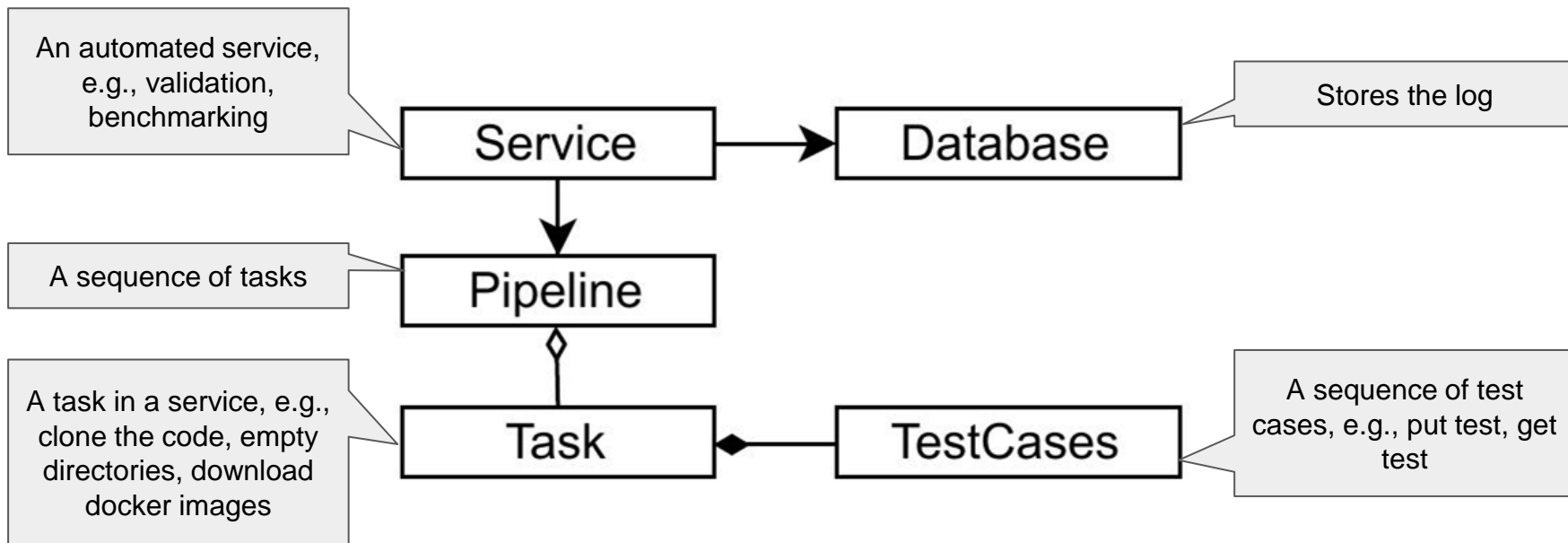
- Gamification with a leaderboard

Framework

Deployment

Web UI

pTA - Framework

A programmable framework to define evaluation tasks

An automated service, e.g., validation, benchmarking → Service

Stores the log → Database

A sequence of tasks → Pipeline

A task in a service, e.g., clone the code, empty directories, download docker images → Task

A sequence of test cases, e.g., put test, get test → TestCases

pTA - Infrastructure

pTA - Web UI (Admin)

pTA - Web UI (Students I)

pTA - Web UI (Students II)

```
Trying to connect to the KVServer...
Succesfully connect to the KVServer.
Checking if there is welcome message on 2 clients...
-------Output@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743>
-------Output@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743>
Testing getting non existent key...
-------Input@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743> get apple123@#
-------Output@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743> Connected successfully
get_error apple123@#

Testing if put works correctly... Client 1 puts
-------Input@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743> put apple123@# orange-@+$
-------Output@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743> put_success apple123@#

Client 2 now read from the db-------Input@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743> get apple123@#
-------Output@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743> Connected successfully
get_success apple123@# orange-@+$

Testing if put works correctly... Client 1 puts
-------Input@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743> put dog one two three
-------Output@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743> put_success dog

Client 2 now read from the db-------Input@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743> get dog
-------Output@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743> get_success dog one two three

Deleting a non existent key...-------Input@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743> delete universe
-------Output@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743> delete_error universe

Deleting a the key from previous test-------Input@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743> delete apple123@#
-------Output@KV at dind-statefulset-0.dind-svc/10.1.241.194:34743> delete_error apple123@#

Test failed. The output does not meet expectation.
```

pTA - Web UI (Leaderboard)

LEADERBOARD: MILESTONE 2

GET	PUT	UPDATE	DELETE

#	Group Name	Latency Score, ms/100op	Throughput Score, 100op/s	Timestamp
1	Group 6	63	15.87	2023-02-11 02:57:09
2	Group 5	79	12.66	2023-02-11 02:56:43
3	Group 2	80	12.50	2023-02-11 02:55:02

pTA in action at TUM and UoT

- pTA has been used at two different universities (TUM and UoT) for two different lab courses
  - 5 semesters at TUM
  - 1 semester at UoT

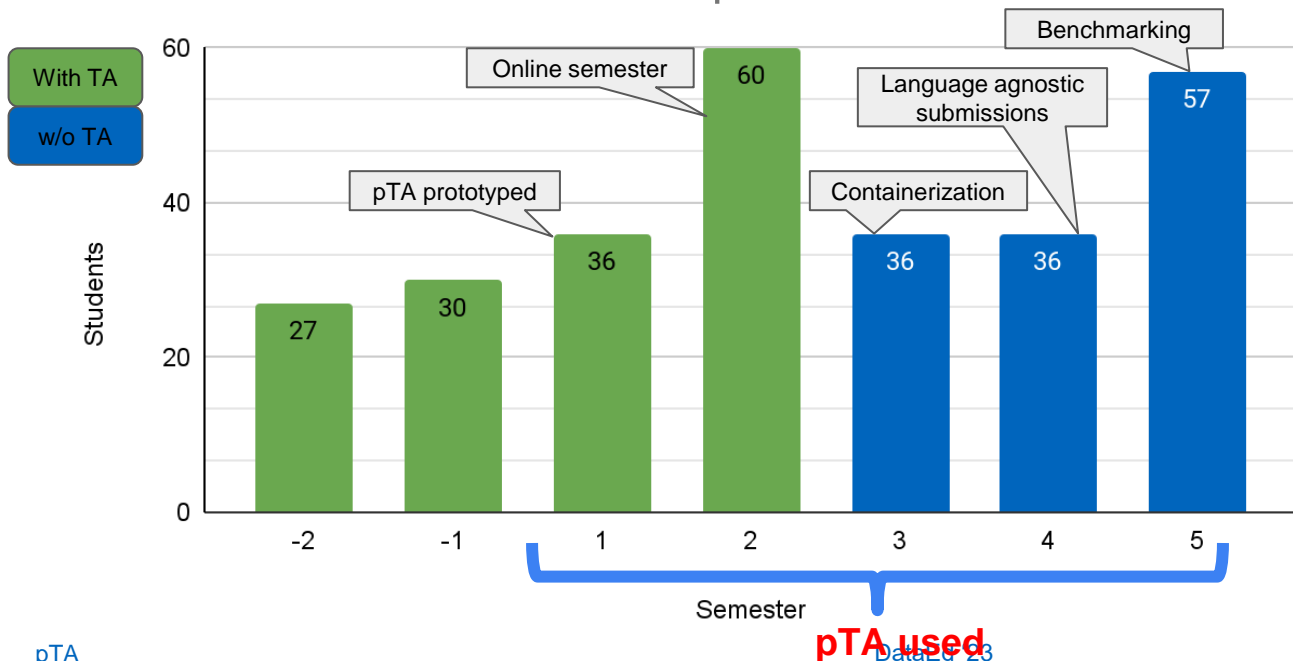- Both courses differ in milestones and submission format

Download docker images
Execute TestCases

TUM Tasks
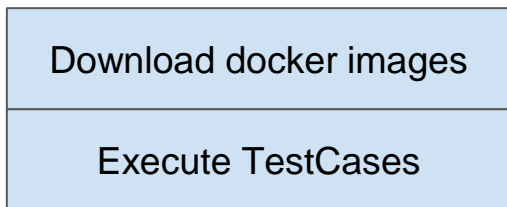
Checkout source code
Build source code
Run JARs
Execute TestCases
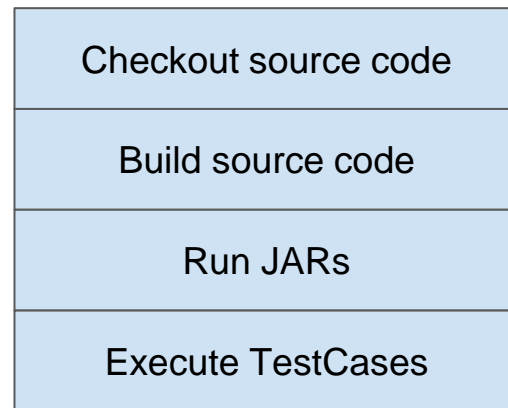
UofT Tasks

pTA makes teaching more efficient.

Students intake per semester. Positive semesters indicates the use of pTA
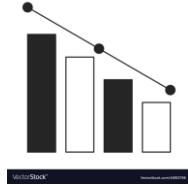
- **2X** course capacity
  - 30 => 57
- Reduced operational cost
  - No more TAs
- Reduced workload
  - Logs => oracle
  - Reduced LMS activity by **75%**
- Pandemic-proof
- Reduced barrier to entry
  - Eliminated programming language dependency
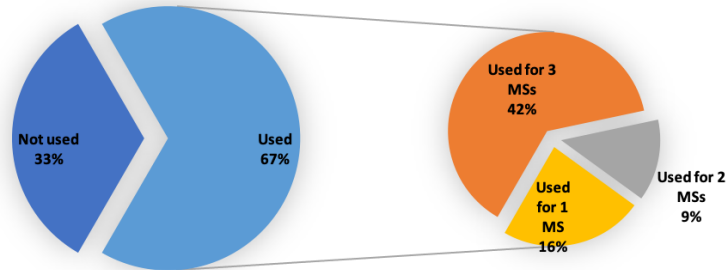
pTA @ UofT

Experimental feature

Decreasing bonus

Manual grading

pTA @ UofT

2/3 of the students used the system and a majority of them kept using the system

Not used 33%

Used 67%

Used for 3 MSs 42%

Used for 2 MSs 9%

Used for 1 MS 16%

Snap evaluation

- Manual evaluations can take up to 40 minutes
- Took 18-36 TA hours to evaluate one MS for all students (n=135)
- pTA performed all evaluations in less than an hour

Higher engagement

- Even with decreasing bonuses, students kept using the system
- Conducted a survey, 91% of users said pTA helped them understand the learning outcomes while a 100% of users were in favor of using pTA in subsequent semesters

Conclusion

- pTA is a framework to automate the evaluations of student submissions

- Deployed at TUM and UofT

- **Reduced** the workload of course organizers

- **Reduced** the operational cost of the course

- Increased the course capacity

- Increased the engagement of the students

- Augmented the teaching by providing 24/7 feedback to students

Questions? Comments? Feedback? Collaborations?

Reach out to us.